USERID ORIGIN	ME1	ME1
DISTRIBUTION CODE	ME436ME	
SPOOL FILE NAME TYPE	REX	PRINTCC
CREATION DATE	06/14/81 10:49:27	
SPOOL FILE ID	0220	
RECORD COUNT	2374	

```
      VV      VV    MM        MM
      VV      VV    MMM      MMM
      VV      VV    MMMM    MMMM
      VV      VV    MM MM  MM MM
 3333333333    77777777777MMMM  00000000
333333333333    77777777777  MM  0000000000
33      VV33    77VV    77      00MM      00
        V33     VV     77M      00MM      00
         33     VV     77MM     00MM      00
       3333VV   VV     77 MM    00MM      00
       3333 VVVV      77 MM    00MM      00
         33 VV        77 MM    00MM      00
         33           77       00        00
33       33           77       00        00
333333333333          77        0000000000
 3333333333           77         00000000
```

```
MM          MM  EEEEEEEEEEEE      11
MMM        MMM  EEEEEEEEEEEE     111
MMMM      MMMM  EE               1111
MM MM    MM MM  EE                11
MM  MMMM  MM    EE                11
MM   MM   MM    EEEEEEEE          11
MM        MM    EEEEEEEE          11
MM        MM    EE                11
MM        MM    EE                11
MM        MM    EE                11
MM        MM    EEEEEEEEEEEE  1111111111
MM        MM    EEEEEEEEEEEE  1111111111
```

```
MM          MM  EEEEEEEEEEEE
MMM        MMM  EEEEEEEEEEEE
MMMM      MMMM  EE
MM MM    MM MM  EE
MM  MMMM  MM    EE
MM   MM   MM    EEEEEEEE
MM        MM    EEEEEEEE
MM        MM    EE
MM        MM    EE
MM        MM    EE
MM        MM    EEEEEEEEEEEE
MM        MM    EEEEEEEEEEEE
```

```
   444          3333333333     6666666666
  4444         333333333333   666666666666
 44 44        33       33     66      66
44  44               33      66
44  44              33       66
44  44            3333       666666666666
444444444444      3333       666666666666
444444444444             33  66      66
      44                 33  66      66
      44           33    33  66      66
      44           333333333333  666666666666
      44            3333333333    6666666666
      44
      44
```

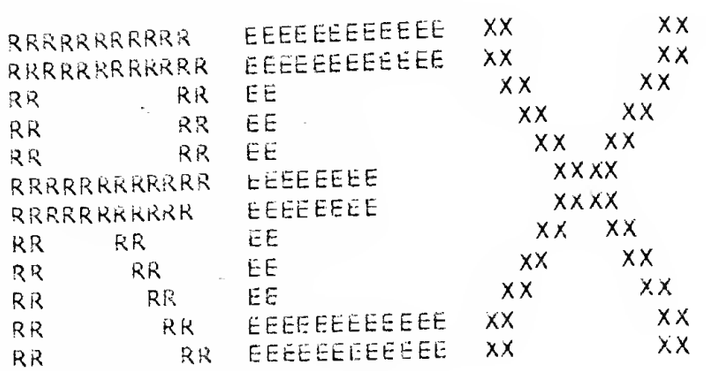
```
MM          MM  EEEEEEEEEEEE
MMM        MMM  EEEEEEEEEEEE
MMMM      MMMM  EE
MM MM    MM MM  EE
MM  MMMM  MM    EE
MM   MM   MM    EEEEEEEE
MM        MM    EEEEEEEE
MM        MM    EE
MM        MM    EE
MM        MM    EE
MM        MM    EEEEEEEEEEEE
MM        MM    EEEEEEEEEEEE
```

USERID ORIGIN	ME1	ME1
DISTRIBUTION CODE	ME436ME	
SPOOL FILE NAME TYPE	REX	PRINTCC
CREATION DATE	08/14/81	10:49:27
SPOOL FILE ID	0220	
RECORD COUNT	2374	

```
      VV        VV     MM        MM
      VV        VV     MMM       MMM
      VV        VV     MMMM      MMMM
      VV        VV     MM MM  MM MM
 3333333333     77777777777MMMM  000000000
333333333333    77777777777  MM  0000000000
33      VV33    77VV    77       00MM      00
        V33     VV     77M       00MM      00
         33     VV    77MM       00MM      00
       3333VV   VV    77 MM      00MM      00
       3333 VVVV     77 MM      00MM      00
         33 VV       77 MM      00MM      00
         33          77         00        00
33       33          77         00        00
333333333333         77          0000000000
 3333333333          77           00000000
```

```
MM          MM  EEEEEEEEEEEE        11
MMM        MMM  EEEEEEEEEEEE       111
MMMM      MMMM  EE                 1111
MM MM    MM MM  EE                   11
MM  MMMM  MM    EE                   11
MM   MM   MM    EEEEEEEE             11
MM        MM    EEEEEEEE             11
MM        MM    EE                   11
MM        MM    EE                   11
MM        MM    EE                   11
MM        MM    EEEEEEEEEEE     1111111111
MM        MM    EEEEEEEEEEE     1111111111
```

```
MM          MM  EEEEEEEEEEEE          444       3333333333    6666666666  MM          MM  EEEEEEEEEEEE
MMM        MMM  EEEEEEEEEEEE         4444      333333333333  666666666666 MMM        MMM  EEEEEEEEEEEE
MMMM      MMMM  EE                  44 44      33        33  66        66  MMMM      MMMM  EE
MM MM    MM MM  EE                 44  44                33  66            MM MM    MM MM  EE
MM  MMMM  MM    EE                44   44                33  66            MM  MMMM  MM    EE
MM   MM   MM    EEEEEEEE         4444444444          3333  66666666666   MM   MM   MM    EEEEEEEE
MM        MM    EEEEEEEE         4444444444          3333  666666666666  MM        MM    EEEEEEEE
MM        MM    EE                    44                33  66        66  MM        MM    EE
MM        MM    EE                    44                33  66        66  MM        MM    EE
MM        MM    EE                    44        33      33  66        66  MM        MM    EE
MM        MM    EEEEEEEEEEE           44       333333333333 666666666666 MM        MM    EEEEEEEEEEE
MM        MM    EEEEEEEEEEE           44        3333333333   6666666666  MM        MM    EEEEEEEEEEE
```



A REFORMED EXECUTOR

GENERAL DOCUMENTATION: REX VERSION 2.01.

17TH AUGUST 1980

MIKE COWLISHAW,
MAIL POINT 182,
IBM UK LABS, HURSLEY.
CJN: REXMAIL AT WINPA

```
RRRRRRRRRR    EEEEEEEEEEEE   XX         XX
RRRRRRRRRRR   EEEEEEEEEEEE   XX         XX
RR        RR  EE              XX       XX
RR        RR  EE               XX     XX
RR        RR  EE                XX   XX
RRRRRRRRRRR   EEEEEEEE           XXXX
RRRRRRRRRR    EEEEEEEE           XXXX
RR    RR      EE                XX   XX
RR     RR     EE               XX     XX
RR      RR    EE              XX       XX
RR       RR   EEEEEEEEEEEE   XX         XX
RR        RR  EEEEEEEEEEEE   XX         XX
```

A REFORMED EXECUTOR

GENERAL DOCUMENTATION: REX VERSION 2.01.

17TH AUGUST 1980

MIKE COWLISHAW,
MAIL POINT 182,
IBM UK LABS, HURSLEY.
CJN: REXMAIL AT WINPA

MAJOR CHANGES AND ENHANCEMENTS FOR REX 2.01 (SINCE I.12)

* CONSIDERABLY ENHANCED DO INSTRUCTION ALLOWS STEPPING OF A VARIABLE DURING A LOOP, USING THE FORM "DO NAME=A TO B BY C;" FOR EXAMPLE. THIS REDUCES LOOP OVERHEAD IN MANY CASES BY A FACTOR OF 4 OR 5. WHILE OR UNTIL MAY BE USED ON ALL FORMS OF DO.

* NEW CONTROL STATEMENTS LEAVE AND ITERATE PERMIT EXIT OR STEPPING OF CONTROLLED DO-LOOPS.

* REXFNS, REXWORDS, AND REXFNS2 WILL BE AUTOMATICALLY LOADED BY REX IF (AND ONLY IF) REQUIRED. REXFNS2 HAS SEVERAL NEW FUNCTIONS. SUBSTR NOW PADS WITH BLANKS, TRANS IS IN REXFNS.

* CLAUSES MAY BE CONTINUED FROM ONE LINE TO ANOTHER USING THE NEW CONTINUATION CHARACTER ",".

* EXTENDED AND REDEFINED ADDRESS INSTRUCTION ALLOWS A SINGLE COMMAND TO BE SENT TO AN ENVIRONMENT. 'CMS' REINSTATED AS NAME FOR HOST.

* LOGICAL EXPRESSIONS MUST RESULT IN '0' OR '1'.

* ENHANCEMENTS TO TRACING:
  INVALID TRACE OPTIONS NOW GIVE A SYNTAX ERROR.
  CALL TRACE-BACK GIVEN IF ERROR OCCURS INSIDE A SUBROUTINE ETC.

* THE DOUBLE-QUOTE MAY BE USED AS AN ALTERNATIVE TO THE SINGLE-QUOTE TO DELIMIT STRINGS. DOUBLE-QUOTE IS THEREFORE NOW A SPECIAL CHARACTER. EG: SAY "ISN'T THIS NICE?"

* VARIABLE NAMES (OTHER THAN ".") MAY NOT BEGIN WITH A PERIOD.

* VALID CHARACTERS FOR SYMBOLS ARE A-Z, a-z, 0-9, @#¢%. ?, UNDERSCORE

* NEW NOP (NO-OPERATION) COMMAND FOR DUMMY THEN/ELSE/WHEN STATEMENTS.

* RC AFTER 'SIGNAL ON SYNTAX' DOES NOT HAVE 20000 ADDED.

* MANY LENGTH AND FUNCTION RESTRICTIONS SOMEWHAT RELAXED.
  (EG, CLAUSES MAY NOW BE UP TO 500 CHARACTERS LONG).

* A CLAUSE OF THE FORM "SYMBOL=EXPRESSION" IS NOW ALWAYS TAKEN TO BE AN ASSIGNMENT. THIS MAKES KEYWORDS FAR LESS RESERVED.
  THEN NOW ONLY ACTS AS A TERMINATOR FOR IF OR WHEN CLAUSES.

* ONLINE HELP AND TUTORIAL HAS ADDED FUNCTION, ALSO A SUB-INDEX FOR EACH OF IOX, FSX, REXFNS ETC. FSX HAS IMPROVED LIGHTPEN ABILITY.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 WHAT IS REX?

REX IS A COMMAND PROGRAMMING LANGUAGE. IT CAN BE USED AS A DIRECT REPLACEMENT FOR OR ALTERNATIVE TO THE CMS EXEC AND EXEC 2 LANGUAGES, AND AS A "MACRO PROCESSOR" FOR EDITORS ETC.

IT HAS SEVERAL ADVANTAGES OVER THE STANDARD CMS EXEC LANGUAGE: IT DOES NOT TOKENISE DATA; IT IS MORE FLEXIBLE AND GENERAL; AND IT IS VERY MUCH EASIER TO LEARN AND USE. IF THE EXEC INVOLVES LOOPS OF ANY KIND, THEN REX IS SIGNIFICANTLY FASTER.

COMPARED WITH EXEC 2, REX HAS SUPERIOR CONTROL STRUCTURES AND EXPRESSION EVALUATION, AND IN MANY WAYS IS MORE SUITABLE FOR EDITOR MACRUS AND COMMAND PARSING.

THE LANGUAGE ITSELF IS PL/I LIKE, AND LARGELY SYSTEM INDEPENDENT. IN ITS CMS IMPLEMENTATION IT IS EASILY INSTALLED AS A "NUCLEUS EXTENSION", EITHER UNDER ITS OWN NAME OR MORE USEFULLY UNDER THE NAME 'EXEC'. IN THIS MODE YOU MAY WRITE FILES WITH FILETYPE EXEC IN ONE OF THREE LANGUAGES: EXEC (STANDARD CMS), EXEC 2 (ITS REPLACEMENT), OR REX.

THE REX INTERFACE WILL EXAMINE THE FILE AND PASS IT ON TO THE APPROPRIATE INTERPRETER. (IF THE FILE BEGINS WITH A REX COMMENT IT WILL BE INTERPRETED BY REX, ETC.). THIS MEANS THAT REX WILL COEXIST WITH BOTH EXEC AND EXEC 2 AND YOU MAY GRADUALLY CONVERT TO REX WITHOUT HAVING TO CHANGE ANY OF YOUR EXISTING EXECS. (SEE THE SECTION ON 'SYSTEM INTERFACES' FOR DETAILS). YOU MAY, TOO, INVOKE THE REX INTERPRETER FROM A PROGRAM WITH THE DATA TO BE INTERPRETED HELD IN STORAGE, SO AVOIDING FILE SYSTEM OVERHEADS.

1.2 WHY REX WAS DESIGNED

THE CMS EXEC LANGUAGE (WHICH HAS SINCE BEEN EXTENDED AND IMPROVED UPON BY EXEC 2) IS BASED ON THE COMMON MACRO LANGUAGE PRINCIPLE THAT VARIABLES AND CONTROLS SHOULD BE DISTINGUISHED (BY "&") AND LITERALS SHOULD EXIST IN 'PLAIN TEXT'.

WHEN EXECS CONSISTED MAINLY OF STRINGS OF COMMANDS, WITH VERY LITTLE LOGIC IN BETWEEN, THIS WAS A FAIR AND SENSIBLE CHOICE: HOWEVER A QUICK SCAN THROUGH THE EXECS OF ALMOST ANY MODERN USER QUICKLY SHOWS THAT THE MAJORITY OF WORDS IN USE ARE SYMBOLIC (THAT IS, THEY BEGIN WITH "&"). THIS OBSERVATION MUST CAST SOME DOUBT ON THE VALIDITY OF USING THIS SYNTAX.

A FURTHER ARGUMENT IS THE INCREASING USE OF "COMPLICATED" STRINGS IN EXECS: FOR EXAMPLE EMBEDDED BLANKS ARE HEAVILY USED IN EDITOR MACROS; FULL SCREEN DISPLAYS; AND SO ON. EXEC 2 HANDLES THESE ONLY FAIRLY WELL, WHEREAS EXEC CANNOT MANIPULATE THEM AT ALL: THE USER IS REDUCED TO UNREADABLE MANIPULATIONS OF THE UNDERSCORE CHARACTER OR OTHER MACHINATIONS TO ACHIEVE THE DESIRED RESULT.

THERE IS PERHAPS AT LEAST SOME JUSTIFICATION IN INVESTIGATING AN ALTERNATIVE MACRO COMMAND LANGUAGE WHICH USES THE 'MORE CONVENTIONAL'

NOTATION USED BY THE HIGHER LEVEL PROGRAMMING LANGUAGES SUCH AS PL/I; PL/S; PASCAL; AND SO ON.

EXPERIENCE SUGGESTS THAT A LANGUAGE WITH THIS TYPE OF SYNTAX WILL BE EASIER TO LEARN AND USE THAN THAT WHICH MORE RESEMBLES A 'MACRO LANGUAGE'. PL/I USERS IN PARTICULAR FIND REX ESPECIALLY USEFUL, AND MANY PEOPLE WHO BEFORE WOULD NOT LEARN A COMMAND LANGUAGE NOW USE REX.

THE USE OF THIS TYPE OF NOTATION WILL NATURALLY CAUSE USERS TO DRAW COMPARISONS WITH THE NORMAL PROGRAMMING LANGUAGES. THIS INEVITABLY WILL LEAD THEM TO EXPECT A CORRESPONDING IMPROVEMENT IN THE FACILITIES AVAILABLE IN THE COMMAND LANGUAGE: THIS IN TURN WOULD SEEM TO IMPLY THAT THE INTERPRETER MIGHT BE LARGER AND PROBABLY SLOWER THAN EITHER EXEC OR EXEC 2. SIZE (WITHIN REASON) IS NOT OFTEN A PROBLEM ON MODERN VIRTUAL MACHINES, HOWEVER A SEVERE PERFORMANCE PENALTY WOULD BE UNACCEPTABLE IN SOME ENVIRONMENTS. SOME EFFORT HAS BEEN MADE TO ENSURE GOOD PERFORMANCE.

DURING IMPLEMENTATION IT HAS BEEN FOUND THAT REX IS OF SIMILAR SIZE TO THE EXISTING INTERPRETERS (CURRENTLY ABOUT 16000 BYTES, 10% OF WHICH ARE THE ERROR MESSAGES). THE CURRENT VERSION IS NORMALLY SOMEWHAT SLOWER THAN EXEC 2; BUT USUALLY VERY MUCH FASTER THAN EXEC.

WHAT THEN ARE THE MAJOR DESIRABLE FEATURES FOR A COMMAND MACRO LANGUAGE? MY CHOICE INCLUDED:

1) STRUCTURED FLOW CONTROL STATEMENTS, SOME EQUIVALENT OF IF-THEN-ELSE, DO (UNTIL/WHILE)-END, SELECT-WHEN-END BEING THE MOST IMPORTANT.
2) FREE FORMAT: NOT LINE-BY-LINE
3) SOME AUTOMATIC CASE TRANSLATION AS REQUIRED BY THE HOST, AND NO UPPER CASE REQUIREMENTS FOR MACRO KEYWORDS ETC.
4) LITERAL SHORTHAND: UNKNOWN "TOKENS" ASSUMED TO BE ENCLOSED IN QUOTES.
5) 'COMPLEX' EXPRESSIONS (I.E. PARENTHESES, MULTIPLE OPERATORS)
6) IN LINE 'FUNCTION' CALLS TO OTHER EXECS OR MODULES
7) NO REQUIREMENT FOR SELF-MODIFYING EXECS
8) "PEER" COMMUNICATION BETWEEN EXECS AND PROGRAMS

THE NEXT SECTION DISCUSSES THESE TOPICS IN MORE DETAIL, HOWEVER THE BUSY (OR IMPATIENT) READER MAY PREFER TO SKIP TO THE LANGUAGE DEFINITION IN SECTION 3.

2. THE LANGUAGE FEATURES

THE FOLLOWING ITEMS ARE NOT INTENDED TO BE RIGOROUS DEFINITIONS OF THE LANGUAGE FEATURES; AND SOME IMPLICIT ASSUMPTIONS ABOUT THE LANGUAGE SYNTAX AND THE HOST SYSTEM WILL BE APPARENT. THEY ARE RATHER GENERAL DESCRIPTIONS OF THE SYNTAX AND THE DECISIONS LEADING TO EACH CHOICE.

2.1 STRUCTURED FLOW CONTROL STATEMENTS

THE NEED FOR STRUCTURED FLOW CONTROL IS ACCEPTED BY MOST PROGRAMMERS. THE THREE MAIN CLASSES OF STRUCTURED FLOW CONTROL ARE THE IF-THEN-ELSE; DO - (ITERATION/WHILE/UNTIL)-END; AND SELECT-WHEN-END. (THE USE OF IBM (PL/I) CONSTRUCTIONS RATHER THAN ANY OF THE POSSIBLY SUPERIOR ALTERNATIVES DESCRIBED IN THE LITERATURE IS PURELY FOR CONSISTENCY.) IF-THEN-ELSE HAS BEEN IMPLEMENTED FOR EXEC BY USING MODULES; EXEC 2 HAS DO-WHILE AND DO-UNTIL; BUT NEITHER HAS ANY FORM OF SELECT (OR CASE) STRUCTURE. EVIDENTLY ALL THESE FEATURES ARE HIGHLY DESIRABLE FOR ANY MODERN LANGUAGE, EVEN IF IN A SIMPLIFIED FORM.

2.2 FREE FORMAT: NOT LINE-BY-LINE

A FREE FORMAT STATEMENT IS POSSIBLY MORE GENERAL THAN FIXED (LINE-BY-LINE) FORMAT. THE LATTER OPTION IMPLIES A RECORD ORIENTED FILE SYSTEM, WHEREAS THE FORMER IS APPLICABLE BOTH TO RECORD AND CHARACTER STREAM FILES OR INPUT DEVICES. BY THE SAME TOKEN, A FREE FORMAT STRUCTURE WOULD IN GENERAL PERMIT BETTER SELF-DOCUMENTATION OF EXECS, SINCE COMMENTS MAY OCCUR ALMOST ANYWHERE IN THE INPUT STREAM.

ALTHOUGH THE LANGUAGE IS BY NATURE AND SYNTAX A STREAM LANGUAGE, MOST IBM USERS WILL TEND TO USE A LINE-BY-LINE FORMAT, WITH ONLY A FEW MULTI-STATEMENT LINES. THEREFORE REX TERMINATES EACH LINE (EXCEPT WHEN WITHIN A STRING OR COMMENT, OR WHEN INHIBITED BY THE CONTINUATION CHARACTER ",") WITH AN IMPLICIT CLAUSE DELIMITER AS A SERVICE TO THE USER. SINCE REX IS AWARE OF LINE ENDS IT CAN INDICATE THE LINE NUMBER IN ERROR MESSAGES.

THE OBVIOUS CLAUSE DELIMITER TO USE WAS ';', WITH /*...*/ FOR COMMENTS.

2.3 CASE TRANSLATION

THE OPERATING SYSTEM UNDERLYING THE REX INTERPRETER MAY REQUIRE THAT COMMANDS BE PASSED IN UPPER CASE. IN THE CMS/CP ENVIRONMENT THERE IS HOWEVER NO SIMPLE RULE FOR DETERMINING WHEN THIS TRANSLATION SHOULD TAKE PLACE: THE ONUS IS UNFORTUNATELY ON THE USER.

REX ATTEMPTS TO MAKE THIS AS EASY AS POSSIBLE BY TRANSLATING ALL SYMBOLS (AS OPPOSED TO STRINGS) TO UPPER CASE DURING SCANNING. THIS HAS TWO MAIN EFFECTS: A) NO REX KEYWORDS NEED BE TYPED IN UPPER CASE: ANY MIXTURE OF UPPER AND LOWER CASE IS VALID; AND B) MOST SYMBOLIC DATA (SEE NEXT SECTION) PASSED TO THE HOST WILL BE AUTOMATICALLY TRANSLATED AND THEREFORE ALSO MAY BE ENTERED IN MULTI-CASE. MULTI-CASE PROGRAMS ARE OF COURSE MORE READABLE AND LESS PRONE TO HAVE ERRORS AND BUGS, SINCE WE ALL ARE TRAINED IN READING LOWER CASE CHARACTERS AS STANDARD.

NOTE THAT DATA IN A STRING OR VARIABLE WILL NEVER BE ALTERED WITHOUT THE EXPRESS INTENTION OF THE USER.

2.4 LITERAL SHORTHAND & BLANK OPERATOR

A CONVENIENT CONVENTION FOR A COMMAND LANGUAGE IS THAT OF 'LITERAL SHORTHAND'. MY DEFINITION OF THIS IS: IF A SYMBOL IS UNKNOWN (I.E. NOT A VARIABLE, REX KEYWORD, OR FUNCTION CALL) THEN IT IS ASSUMED TO REPRESENT A LITERAL STRING CONSISTING OF THE CHARACTERS OF THE SYMBOL (TRANSLATED TO UPPER CASE).

A FURTHER CONVENIENCE IS THE CONCEPT OF THE 'BLANK' OPERATOR. THIS MAY BE DEFINED VERBALLY THUS: IF TWO EXPRESSIONS (IE SYMBOLS, LITERALS, ETC.) ARE SEPARATED BY ONE OR MORE BLANKS AND NO OTHER OPERATOR THEN THE OPERATION OF 'CONCATENATE WITH A BLANK IN BETWEEN' WILL BE PERFORMED.

THE EFFECT OF THESE TWO CONVENTIONS ALLOWS A SYNTAX THAT COMBINES THE ADVANTAGES OF BOTH EXEC/MACRO LANGUAGES AND THE PL/I LIKE MODEL. CONSIDER THE FOLLOWING EXCERPT FROM A REX EXEC (ASSUME THAT FN, FT, FM ARE SYMBOLS REPRESENTING VARIABLES PREVIOUSLY SET UP BY ASSIGNMENTS ETC.):

```
STATE FN FT FM
IF RC=0 THEN ERASE FN FT FM
```

WHICH IS MORE READABLE THAN THE EQUIVALENT 'STRICT PL/I' FORM:

```
'STATE '||FN||' '||FT||' '||FM;
IF RC=0 THEN 'ERASE '||FN||' '||FT||' '||FM;
```

OR THE EXEC LANGUAGE FORM:

```
STATE &FN &FT &FM
&IF &RETCODE = 0 ERASE &FN &FT &FM
```

(A STATEMENT WHICH IS AN EXPRESSION ON ITS OWN IS PASSED TO THE HOST SYSTEM AS A COMMAND.)

2.5 'COMPLEX' EXPRESSIONS

COMPOUND CHARACTER AND ARITHMETIC EXPRESSIONS ARE BEING USED MORE AND MORE IN CURRENT EXECS: THEY DO OF COURSE HAVE TO BE SPREAD OVER SEVERAL LINES. (UP TO TEN LINES FOR ONE LOGICAL MANIPULATION IS NOT UNKNOWN). REX THEREFORE PERMITS 'COMPLEX' EXPRESSIONS.

THERE ARE THREE POSSIBLE IMPLEMENTATIONS OF COMPOUND EXPRESSIONS:

A) SIMPLE LEFT -> RIGHT (OR APL RIGHT -> LEFT) SCANNING;
B) REVERSE POLISH NOTATION (E.G. FORTH);
C) FULL ALGEBRAIC, WITH PARENTHESES AND OPERATOR PRIORITIES.

OPTION B) IS PROBABLY UNACCEPTABLE TO THE IBM USER, AND IS ALSO SOMEWHAT OUTDATED AS A SOLUTION. OPTION A) IS A CONSIDERABLE IMPROVEMENT ON NO COMPOUND EXPRESSIONS AT ALL, BUT IS NOT IDEAL - ESPECIALLY AS LOGICAL OPERATIONS SHOULD BE TREATED AS NORMAL OPERATORS, RATHER THAN SPECIAL CASES.

OPTION C) IS OF COURSE THE BEST, AND IS NOT SIGNIFICANTLY MORE COMPLICATED TO IMPLEMENT THAN A). THE ALGORITHMS AND TECHNIQUES ARE WELL UNDERSTOOD, AND AN EXEC INTERPRETER NECESSARILY INCLUDES STORAGE MANAGEMENT ROUTINES WHICH NORMALLY ARE ABLE TO HANDLE STACK(S).

I CONSIDER THE MINIMUM SET OF PRIMITIVE DYADIC OPERATORS TO INCLUDE: + - * / || AND ' ' AS DEFINED ABOVE, TOGETHER WITH THE LOGICAL OPERATORS = ¬= > < >= <= & | &&. IMPORTANT MONADIC OPERATORS ARE: ¬ - + (PREFIX NOT, MINUS, AND +).

"(" AND ")" HAVE SPECIAL RULES AFFECTING THEIR USE, SINCE IN ADDITION TO FORCING PRIORITIES WITHIN EXPRESSION EVALUATION, THEY ARE ALSO USED FOR THE INVOCATION OF FUNCTIONS. THEREFORE BLANKS IMMEDIATELY OUTSIDE THE PARENTHESES ARE NOT IGNORED, AND SO THE ' ' OPERATOR MAY ACT DIRECTLY ON A BRACKETED SUB-EXPRESSION.

2.6 IN LINE 'FUNCTION' CALLS TO OTHER EXECS OR MODULES

THE ABILITY TO DEFINE IN-EXPRESSION FUNCTIONS GREATLY INCREASES THE POWER OF A LANGUAGE, AND REDUCES THE NEED FOR SPECIALISED BUILTIN FUNCTIONS.

THE HOST SYSTEM IS ASSUMED TO INCLUDE AT LEAST ONE COMMAND EXECUTOR AND SOME STORAGE ALLOCATION ROUTINES. A SUB-CLASS OF COMMANDS ARE THOSE WHICH ACCEPT DATA AND/OR ARGUMENTS FROM REX, AND RETURN THEIR RESULT IN A STORAGE BLOCK WHICH IS USABLE BY REX. THIS SUBCLASS CAN BE TERMED 'FUNCTIONS' AND INCLUDED IN THE REX

LANGUAGE USING THE CONVENTIONAL NOTATION OF PARENTHESES, WITH COMMAS TO SEPARATE THE ARGUMENT EXPRESSIONS.

FOR EXAMPLE THE FUNCTION 'SUBSTR' COULD BE IMPLEMENTED BY A SEPARATE MODULE.

THE SYNTAX DESCRIPTION WOULD THEREFORE BE: IF A SYMBOL IS FOLLOWED IMMEDIATELY BY A "(" THEN ITS VALUE (WHICH, RECALL, WILL BE ITS NAME UNLESS IT HAS BEEN USED AS A VARIABLE) IS TAKEN TO BE A FUNCTION NAME. EACH EXPRESSION FOLLOWING THE "(" AND SEPARATED BY "," IS EVALUATED, AND THE FUNCTION IS INVOKED WHEN THE FINAL ")" IS INTERPRETED.

THIS GIVES A 'NORMAL' SYNTAX FOR FUNCTION CALLS, WITHOUT THE NEED FOR A NEW CLAUSE FOR EVERY COMMAND.

THREE PACKS OF USEFUL BASIC FUNCTIONS ARE SUPPLIED WITH THE CMS VERSION OF REX: THESE WILL BE LOADED AUTOMATICALLY IF ANY FUNCTION CONTAINED IN THEM IS INVOKED.

2.7 NO REQUIREMENT FOR SELF-MODIFYING EXECS

EXEC AND EXEC 2 BOTH PERMIT SELF-MODIFYING EXECS. THIS IS A "NICE" FACILITY WHICH HOWEVER IS TYPICALLY NOT USED. IN FACT, THE ONLY TIME IT NORMALLY OCCURS IS WHEN ONE EDITS AN 'EDIT' EXEC: AND THEN IT IS USUALLY MORE OF AN EMBARRASSMENT THAN A HELP.

REX THEREFORE ASSUMES THAT ALL EXECS ARE READ ONLY. THIS IMPLIES THAT: A) THE ENTIRE EXEC IS READ INITIALLY (INEFFICIENT FOR LONG FILES, PERHAPS); AND B) STATEMENTS THAT MIGHT BE RE-INTERPRETED (E.G. IN LOOPS) NEED ONLY BE PARSED ONCE, FOR IMPROVED PERFORMANCE.

IN ADDITION, IT CAN INTERPRET DATA DIRECTLY FROM STORAGE: SO AVOIDING THE OVERHEAD OF LOADING ALL PROGRAMS (EXECS) FROM DISK.

THE 'READ/ONLY' RESTRICTION ALSO OPENS UP THE ATTRACTIVE POSSIBILITY OF COMPILATION OR PART COMPILATION OF THE LANGUAGE: A POSSIBLE IMPLEMENTATION MIGHT THEREFORE CONSIST OF A 'COMPILER' WHICH PRODUCES AN 'OBJECT FILE' WHICH COULD THEN BE VERY EFFICIENTLY INTERPRETED BY THE REX EXEC PROCESSOR, WITH REAL PERFORMANCE IMPROVEMENTS (A FACTOR OF AT LEAST 4 MIGHT BE EXPECTED). HOWEVER, THERE IS AN IDENTIFIABLE NEED FOR THE 'FULLY INTERPRETIVE' METHOD OF EXECUTION, AND THIS HAS BEEN IMPLEMENTED FIRST.

2.8 "PEER" EXEC/PROGRAM COMMUNICATION

IT IS OFTEN DESIRABLE TO SUSPEND THE EXECUTION OF AN EXEC IN ORDER TO CARRY ON A DIALOGUE WITH ANOTHER EXEC OR PROGRAM, WITHOUT HAVING TO ENTER THE EXEC 'AT THE TOP' FOR EACH INVOCATION. AN OBVIOUS EXAMPLE OF THIS IS EDITOR MACROS, WHERE THE EXEC NEEDS TO GET ADDITIONAL OR FEEDBACK INFORMATION FROM THE CALLER.

THE YKTSVC CMS PACKAGE IMPLEMENTS AN EFFECTIVE 'SUBCOMMAND' HANDLER, AND IT SEEMS SENSIBLE AND PRACTICAL TO USE THIS SYSTEM.

ONE REX INSTRUCTION IS USED TO CONTROL THE FACILITY: 'ADDRESS NNN' WILL CAUSE ANY FOLLOWING COMMANDS TO BE ROUTED TO THE ENVIRONMENT NAMED NNN, AND 'ADDRESS' (NO NAME) WILL RE-ROUTE ALL FOLLOWING COMMANDS TO THE PREVIOUSLY SELECTED ENVIRONMENT. SIMILARLY 'ADDRESS NNN EXPRESSION' WILL SEND JUST ONE COMMAND TO THE IDENTIFIED ENVIRONMENT.

REX IS FULLY COMPATIBLE WITH EXEC 2, AND PROGRAMS WHICH SUCCESSFULLY INTERFACE WITH EXEC 2 SHOULD BE ABLE TO USE REX WITHOUT ANY CHANGES BEING NECESSARY. THIS HAS OBVIOUS ADVANTAGES.

3. REX LANGUAGE DEFINITION

LANGUAGE DEFINITION FOR REX VERSION 2.01.

NOTE: THIS DEFINITION ATTEMPTS TO BE A COMPLETE DESCRIPTION OF THE SYNTAX, WHICH IS NOW "FROZEN" IN THE SENSE THAT INCOMPATIBLE CHANGES WILL NOT BE MADE EXCEPT IN EXTRA-ORDINARY CIRCUMSTANCES. PLEASE BRING ANY ERRORS, OMISSIONS, OR NECESSARY CLARIFICATIONS TO THE ATTENTION OF THE AUTHOR: SEE ADDRESS ON THE FRONT OF THIS DOCUMENT.

3.1 STRUCTURE AND GENERAL SYNTAX

A REX PROGRAM IS BUILT UP OUT OF A SERIES OF 'CLAUSES' WHICH ARE COMPOSED OF: ZERO OR MORE BLANKS (WHICH ARE IGNORED); A SEQUENCE OF TOKENS (SEE BELOW); ZERO OR MORE BLANKS (AGAIN IGNORED); AND THE DELIMITER ";" (SEMICOLON) WHICH MAY BE IMPLIED BY LINE-END, CERTAIN KEYWORDS, OR THE COLON ":". EACH CLAUSE IS SCANNED BEFORE EXECUTION FROM LEFT TO RIGHT AND THE TOKENS COMPOSING IT ARE IDENTIFIED. COMMENTS ARE REMOVED, AND MULTIPLE BLANKS (EXCEPT WITHIN STRINGS) ARE CONVERTED TO SINGLE BLANKS. BLANKS ADJACENT TO SPECIAL CHARACTERS (SEE BELOW) ARE ALSO REMOVED.

3.1.1 TOKENS

THE LANGUAGE IS COMPOSED OF TOKENS (OF ANY LENGTH, UP TO AN IMPLEMENTATION RESTRICTED MAXIMUM) WHICH ARE SEPARATED BY BLANKS OR BY THE NATURE OF THE TOKENS THEMSELVES. THE CLASSES OF TOKENS ARE:

COMMENTS:

ANY SEQUENCE OF CHARACTERS ON ONE OR MORE LINES WHICH ARE DELIMITED BY '/*' AND '*/'. COMMENTS MAY BE NESTED, WHICH IS TO SAY THAT '/*' AND '*/' MUST PAIR CORRECTLY. COMMENTS ARE IGNORED BY THE INTERPRETER (AND HENCE MAY BE OF ANY LENGTH), BUT DO ACT AS SEPARATORS.

STRINGS:

A STRING INCLUDING ANY CHARACTERS AND DELIMITED BY THE SINGLE QUOTE CHARACTER (') OR THE DOUBLE-QUOTE ("). USE "" TO INCLUDE A " IN A STRING DELIMITED BY ", AND USE '' TO INCLUDE A ' IN A STRING DELIMITED BY '. A STRING IS A LITERAL CONSTANT AND ITS CONTENTS WILL NEVER BE MODIFIED BY REX. NOTE THAT IF FOLLOWED IMMEDIATELY BY A "(", THE STRING WILL BE TAKEN TO BE THE NAME OF A FUNCTION; AND IF FOLLOWED IMMEDIATELY BY AN 'X' THEN IT SHOULD BE A HEX STRING:

HEX STRINGS:

ANY SEQUENCE OF PAIRS OF HEXADECIMAL DIGITS (0-9, A-F, A-F) OPTIONALLY SEPARATED BY BLANKS, DELIMITED BY SINGLE- OR DOUBLE- QUOTES AND IMMEDIATELY FOLLOWED BY THE CHARACTER 'X' OR 'X'. THIS REPRESENTS A CHARACTER STRING CONSTANT FORMED BY PACKING THE HEXADECIMAL CODES GIVEN. THE BLANKS, WHICH MAY ONLY BE PRESENT AT BYTE BOUNDARIES, ARE TO AID READABILITY AND ARE ENTIRELY IGNORED.

SYMBOLS:

GROUPS OF ANY EBCDIC CHARACTERS, SELECTED FROM THE ALPHABETIC AND NUMERIC CHARACTERS (A-Z, A-Z, 0-9) AND/OR FROM THE CHARACTERS @#£% . ? AND UNDERSCORE, ARE CALLED SYMBOLS. ANY LOWER CASE ALPHABETIC CHARACTER IN A SYMBOL IS TRANSLATED TO UPPER CASE. IF THE SYMBOL THEN MATCHES A REX KEYWORD AND IT IS AT THE BEGINNING OF A CLAUSE AND IT IS NOT FOLLOWED BY AN "=", THEN IT IS INTERPRETED SPECIALLY. OTHERWISE IF IT DOES NOT BEGIN WITH A DIGIT (0-9) OR A PERIOD (OR IF IT IS JUST A SINGLE PERIOD) THEN IT IS POTENTIALLY A VARIABLE AND MAY HAVE A VALUE. IF IT DOES NOT HAVE A VALUE THEN IT IS INTERPRETED AS THE CHARACTER STRING CONSISTING OF THE CHARACTERS OF THE SYMBOL (AS THOUGH THEY OCCURRED WITHIN QUOTES).

NUMBERS:

THESE ARE CHARACTER STRINGS CONSISTING OF ONE OR MORE DECIMAL DIGITS OPTIONALLY PREFIXED BY A PLUS OR MINUS SIGN, AND OPTIONALLY INCLUDING A SINGLE PERIOD ("."). NUMBERS MAY HAVE LEADING BLANKS (BEFORE AND/OR AFTER THE SIGN, IF ANY) AND MAY HAVE TRAILING BLANKS. EMBEDDED BLANKS ARE NOT PERMITTED. NOTE THAT A SYMBOL (ABOVE) MAY BE A NUMBER AND SO MAY A STRING CONSTANT.

OPERATORS:

THE SPECIAL CHARACTERS: + - / * | & = ¬ > < AND THE SEQUENCES >= <= ¬> ¬< ¬= == // && || ARE OPERATOR TOKENS. ONE OR MORE BLANK CHARACTER(S), WHERE THEY OCCUR IN EXPRESSIONS BUT ARE NOT ADJACENT TO ANOTHER OPERATOR, ALSO ACT AS AN OPERATOR.

SPECIAL CHARACTERS:
THE CHARACTERS , ; :) (' " TOGETHER WITH THE INDIVIDUAL CHARACTERS FROM THE OPERATORS HAVE SPECIAL SIGNIFICANCE WHEN FOUND OUTSIDE STRINGS, AND CONSTITUTE THE SET OF "SPECIAL" CHARACTERS. THEY ALL ACT AS TOKEN DELIMITERS, AND BLANKS ADJACENT TO ANY OF THESE ARE REMOVED, WITH THE EXCEPTION THAT A BLANK ADJACENT TO THE OUTSIDE OF A PARENTHESIS IS ONLY DELETED IF IT IS ALSO ADJACENT TO ANOTHER SPECIAL CHARACTER.

FOR EXAMPLE THE CLAUSE 'REPEAT' B + 3 IS COMPOSED OF FIVE TOKENS: A STRING, A BLANK OPERATOR, A SYMBOL (WHICH MAY HAVE A VALUE), AN OPERATOR, AND A SECOND SYMBOL (WHICH ALSO IS A NUMBER). THE BLANKS BETWEEN THE "B" AND THE "+" AND BETWEEN THE "+" AND THE "3" ARE REMOVED, HOWEVER ONE OF THE BLANKS BETWEEN THE 'REPEAT' AND THE "B" REMAINS AS AN OPERATOR.

3.1.2 IMPLIED SEMICOLONS AND CONTINUATIONS

REX WILL NORMALLY ASSUME (IMPLY) A SEMI-COLON AT THE END OF EACH LINE, EXCEPT IF:

A) THE LINE ENDS IN THE MIDDLE OF A STRING
B) THE LINE ENDS IN THE MIDDLE OF A COMMENT
C) NEITHER OF THE ABOVE CASES HOLD, BUT THE LAST NON-COMMENT TOKEN WAS A COMMA. IN THIS CASE THE COMMA IS FUNCTIONALLY REPLACED BY A BLANK, AND HENCE ACTS AS A 'CONTINUATION CHARACTER'. NOTE THAT THE COMMA WILL REMAIN IN EXECUTION TRACES.

THIS MEANS THAT SEMICOLONS NEED ONLY BE INCLUDED WHEN THERE IS MORE THAN ONE CLAUSE ON A LINE.

(NOTE: THE TWO CHARACTERS FORMING A DOUBLE QUOTE WITHIN A STRING, OR THE COMMENT DELIMITERS "/*" AND "*/" SHOULD NOT BE SPLIT BY A LINE END SINCE THEY COULD NOT THEN BE RECOGNISED CORRECTLY: AN IMPLIED SEMICOLON WOULD BE ADDED.)

(NOTE: SEMI-COLONS ARE ADDED AUTOMATICALLY BY REX AFTER COLONS, AND AFTER CERTAIN KEYWORDS WHEN IN THE CORRECT CONTEXT. THE KEYWORDS THAT MAY HAVE THIS EFFECT ARE: ELSE OTHERWISE THEN. THESE SPECIAL CASES REDUCE TYPOGRAPHICAL ERRORS SIGNIFICANTLY).

3.2 EXPRESSIONS AND OPERATORS

MANY CLAUSES MAY INCLUDE EXPRESSIONS WHICH CAN CONSIST OF "TERMS" (SYMBOLS, STRINGS, OR FUNCTION CALLS), INTERSPERSED WITH OPERATORS AND PARENTHESES.

A STRING, OR ANY SYMBOL WHICH MAY BE A VALID NUMBER (BECAUSE IT STARTS WITH A DIGIT, OR IS A "." FOLLOWED BY SOME OTHER CHARACTERS), IS ALWAYS TAKEN TO BE A LITERAL CONSTANT.

OTHER SYMBOLS MAY BE THE NAME OF A VARIABLE, IN WHICH CASE THEY ARE REPLACED BY THE VALUE OF THAT VARIABLE AS SOON AS THEY ARE NEEDED DURING EVALUATION. OTHERWISE THEY ARE TREATED AS A STRING (NOTE THAT LOWER CASE CHARACTERS WILL HAVE BEEN TRANSLATED TO UPPER CASE). A SYMBOL MAY ALSO BE "COMPOUND" - SEE LATER IN THIS DOCUMENT.

EVALUATION OF AN EXPRESSION IS LEFT TO RIGHT, MODIFIED BY PARENTHESES AND BY OPERATOR PRECEDENCE IN THE USUAL "ALGEBRAIC" MANNER (SEE BELOW). SINCE ALL DATA IS IN THE FORM OF TYPELESS CHARACTER STRINGS, THE RESULT OF ANY EXPRESSION EVALUATION IS ITSELF A CHARACTER STRING. ALL TERMS, DATA, AND RESULTS MAY BE THE "NULL STRING" (A STRING OF LENGTH 0).

THE OPERATORS (EXCEPT THE PREFIX OPERATIONS) ACT ON TWO TERMS, WHICH MAY BE INTERMEDIATE RESULTS OR BRACKETED SUB-EXPRESSIONS. PREFIX OPERATORS ACT ON THE FOLLOWING TERM OR SUB-EXPRESSION. THERE ARE FOUR TYPE OF OPERATOR:

STRING CONCATENATION:

(BLANK)	CONCATENATE TWO TERMS WITH A SINGLE BLANK IN BETWEEN
||	CONCATENATE WITHOUT AN INTERVENING BLANK

CONCATENATION WITHOUT A BLANK MAY BE FORCED BY USING THE || OPERATOR, BUT IT IS USEFUL TO KNOW THAT IF A STRING AND A SYMBOL ARE JUXTAPOSED, THEN THEY WILL BE CONCATENATED DIRECTLY.

E.G: FRED'%' WOULD EVALUATE TO '37.3%' IF THE VARIABLE "FRED" HAD THE VALUE '37.3'.

ARITHMETIC:

+	ADD
-	SUBTRACT
*	MULTIPLY
/	DIVIDE (AS PER 370)
//	DIVIDE AND RETURN THE REMAINDER (NOT MODULO)
PREFIX -	NEGATE THE FOLLOWING TERM (WHICH MUST BE NUMERIC)
PREFIX +	TAKE THE FOLLOWING TERM (WHICH MUST BE NUMERIC) AS IS.

SEE THE SECTION ON "NUMERICS" FOR DETAILS OF ACCURACY AND THE FORMAT OF VALID NUMBERS.

COMPARATIVE:

THE COMPARATIVE OPERATORS RETURN THE VALUE '1' IF THE RESULT OF THE COMPARISON IS TRUE, OR '0' OTHERWISE. IF BOTH THE TERMS INVOLVED ARE NUMERIC, THEN A NUMERIC COMPARISON (IN WHICH LEADING ZEROS ARE IGNORED) IS EFFECTED, OTHERWISE BOTH TERMS ARE TREATED AS CHARACTER STRINGS (THE SHORTER IS PADDED WITH BLANKS ON THE RIGHT). THE "==" OPERATOR MAY BE USED TO TEST FOR AN EXACT MATCH BETWEEN TWO STRINGS - IN THIS CASE NO PADDING OR ZEROS REMOVAL TAKES PLACE.

=	TRUE IF THE TERMS ARE EQUAL (WHEN PADDED WITH BLANKS ETC.)
==	TRUE IF THE TERMS ARE EXACTLY EQUAL (IDENTICAL)
¬=	NOT EQUAL (INVERSE OF =)
>	GREATER THAN
<	LESS THAN
>=, ¬<	GREATER THAN OR EQUAL TO, NOT LESS THAN
<=, ¬>	LESS THAN OR EQUAL TO, NOT GREATER THAN

LOGICAL (BOOLEAN):

A CHARACTER STRING IS TAKEN TO HAVE THE VALUE "FALSE" IF IT IS '0', AND "TRUE" IF IT IS A '1'. THE LOGICAL OPERATORS TAKE ONE OR TWO SUCH VALUES (VALUES OTHER THAN '0' OR '1' ARE NOT ALLOWED) AND RETURN '0' OR '1' AS APPROPRIATE:

&	RETURNS '1' IF BOTH TERMS ARE "TRUE"
|	RETURNS '1' IF EITHER TERM IS "TRUE"
&&	RETURNS '1' IF EITHER (NOT BOTH) IS "TRUE" (EXCLUSIVE OR)
PREFIX ¬	LOGICAL "NOT" (NEGATE: "TRUE" -> '0', "FALSE" -> '1')

OPERATOR PRIORITIES:

EXPRESSION EVALUATION IS MODIFIED BY OPERATOR PRECEDENCE. FOR EXAMPLE, "*" (MULTIPLY) HAS A HIGHER PRIORITY THAN "+" (ADD), SO 3+2*5 WILL EVALUATE TO '13' (RATHER THAN THE '25' WHICH WOULD RESULT IF STRICT LEFT TO RIGHT EVALUATION OCCURRED). THE ORDER OF PRECEDENCE OF THE OPERATORS IS (HIGHEST AT THE TOP):

PREFIX ¬, - AND +	(PREFIX OPS.)
* / //	(MULTIPLY AND DIVIDE)
+ -	(ADD AND SUBTRACT)
" ", ||	(CONCATENATION, WITH OR WITHOUT BLANK)
= == ¬= >= <= ¬> ¬<	(COMPARISON OPERATORS)
&	(AND)
| &&	(OR, EXCLUSIVE OR)

EXAMPLES: SUPPOSE THAT THE FOLLOWING SYMBOLS REPRESENT VARIABLES; WITH VALUES AS SHOWN:

```
A   HAS THE VALUE '3'
DAY HAS THE VALUE 'MONDAY'

THEN:

A+5                 WILL EVALUATE TO '8'
A-4*2                    ==>        '-5'
(A+1)>7                  ==>        '0'              /* FALSE */
' '=='                   ==>        '0'              /* FALSE */
(A+1)*3=12               ==>        '1'              /* TRUE  */
TODAY IS DAY             ==>        'TODAY IS MONDAY'
'IF IT IS' DAY           ==>        'IF IT IS MONDAY'
SUBSTR(DAY,2,3)          ==>        'OND'    /* SUBSTR IS A FUNCTION */
' 'XXX' '                ==>        ' XXX '
```

3.3 CLASSES OF INSTRUCTIONS

THE CLAUSES MAY BE SUBDIVIDED INTO FIVE TYPES:

* A NULL CLAUSE:
  A CLAUSE CONSISTING OF ONLY BLANKS AND/OR COMMENTS, OR THE KEYWORD "THEN" ALONE, IS COMPLETELY IGNORED BY REX (EXCEPT THAT COMMENTS MAY BE TRACED).

* A LABEL:
  A CLAUSE WHICH ENDS IN A COLON IS A LABEL, AND SHOULD CONSIST OF A SINGLE SYMBOL FOLLOWED BY THE COLON. LABELS ARE USED TO IDENTIFY THE TARGETS OF CALL AND SIGNAL INSTRUCTIONS.

* AN ASSIGNMENT (SECTION 3.4):
  A SINGLE CLAUSE WITH THE FORM "SYMBOL=EXPRESSION" THIS GIVES A VARIABLE A (NEW) VALUE.

* A CONTROL INSTRUCTION (SECTION 3.6):
  ONE OR MORE CLAUSES, THE FIRST OF WHICH STARTS WITH A REX KEYWORD WHICH IDENTIFIES THE INSTRUCTION. THESE CONTROL THE EXTERNAL INTERFACES, THE FLOW OF CONTROL, ETC. SOME INSTRUCTIONS MAY INCLUDE NESTED INSTRUCTIONS.

* A COMMAND (SECTION 3.5):
  A SINGLE CLAUSE CONSISTING OF AN EXPRESSION. THE EXPRESSION IS EVALUATED AND PASSED AS A COMMAND STRING TO THE HOST OR OTHER ENVIRONMENT.

3.4 ASSIGNMENT INSTRUCTIONS

ANY CLAUSE OF THE FORM:

"SYMBOL=EXPRESSION"

IS TAKEN TO BE AN ASSIGNMENT STATEMENT.

THE SYMBOL IS ANY VALID COLLECTION OF CHARACTERS (AS DESCRIBED ABOVE) BUT EXCLUDING THOSE BEGINNING WITH A DIGIT (0-9) OR A PERIOD (EXCEPT FOR A PERIOD ON ITS OWN, WHICH CANNOT BE A VALID NUMBER). IT MAY ALSO BE 'COMPOUND' (SEE BELOW). BY BEING THE TARGET OF AN ASSIGNMENT IN THIS MANNER, IT IS CONTEXTUALLY DECLARED AS A VARIABLE: IN OTHER WORDS, IN ALL SUCCEEDING STATEMENTS THIS PARTICULAR COLLECTION OF CHARACTERS WITHIN AN EXPRESSION REPRESENTS A STRING IN STORAGE.

NOTE: SINCE AN EXPRESSION MAY INCLUDE THE OPERATOR '=', AND A STATEMENT MAY CONSIST PURELY OF AN EXPRESSION (SEE NEXT SECTION), THERE IS A POSSIBLE AMBIGUITY HERE. REX THEREFORE TAKES ANY CLAUSE WHICH STARTS WITH A SYMBOL AND WHOSE SECOND TOKEN IS '=' TO BE AN ASSIGNMENT STATEMENT, NOT AN EXPRESSION.

THIS IS NOT A RESTRICTION, SINCE FOR EXAMPLE THE CLAUSE MAY BE EXECUTED AS A COMMAND BY PUTTING A '' BEFORE THE FIRST NAME, OR BY ENCLOSING THE ENTIRE EXPRESSION IN PARENTHESES.

SIMILARLY, IF A PROGRAMMER ACCIDENTALLY USES A REX KEYWORD AS A VARIABLE NAME, THIS SHOULD NOT CAUSE CONFUSION - FOR EXAMPLE THE CLAUSE:

ADDRESS='10 DOWNING STREET'

WOULD BE AN ASSIGNMENT STATEMENT, NOT AN ADDRESS INSTRUCTION.

3.5 COMMANDS TO THE HOST

THE 'HOST SYSTEM' FOR REX IS ASSUMED TO INCLUDE AT LEAST ONE QUEUE ("STACK") CAPABLE OF HANDLING CHARACTER STRINGS, AND AT LEAST ONE ENVIRONMENT FOR EXECUTING COMMANDS. ONE OF THESE ENVIRONMENTS MAY BE SELECTED TO BE THE 'CURRENT' ACTIVE ENVIRONMENT.

EXECUTING COMMANDS USING THE CURRENTLY ADDRESSED ENVIRONMENT MAY BE ACHIEVED USING A STATEMENT OF THE FORM:

"EXPRESSION"

THE EXPRESSION IS EVALUATED, RESULTING IN A CHARACTER STRING (WHICH MAY BE THE NULL STRING). IF THE STRING IS NOT NULL, IT IS THEN PREPARED AS APPROPRIATE AND SUBMITTED TO THE HOST. FOR EXAMPLE, IF THE HOST WERE CMS, THEN THE STRING WOULD BE 8-BYTE TOKENISED AND A NEW-FORM PLIST WOULD BE BUILT.

AS AN EXAMPLE OF HOW A CMS COMMAND MIGHT BE ISSUED, THE SEQUENCE:

```
FN=JACK; FT=RABBIT; FM=A1
STATE FN FT FM
```

WOULD RESULT IN THE PLIST: "STATE JACK RABBIT A1 " BEING SUBMITTED TO CMS. OF COURSE, THE SIMPLE EXPRESSION 'STATE JACK RABBIT A1' WOULD HAVE THE SAME EFFECT IN THIS CASE.

ALL COMMANDS RETURN AN INTEGER RETURNCODE. THIS IS PLACED IN THE VARIABLE 'RC' WHEN THE COMMAND HAS FINISHED EXECUTING. BY CONVENTION, A RETURNCODE OF 0 MEANS 'SUCCESSFUL COMPLETION'.

ALTERNATIVE EXECUTION ENVIRONMENTS MAY BE SELECTED BY THE 'ADDRESS' INSTRUCTION (Q.V.) WHICH MAY ALSO BE USED TO ISSUE COMMANDS.

THE DEFAULT ENVIRONMENT WILL DEPEND ON THE CALLER OF REX: FOR EXAMPLE IF CALLED FROM CMS, THEN THE DEFAULT ENVIRONMENT WOULD BE CMS, IF CALLED PROPERLY FROM AN EDITOR, THEN THE DEFAULT ENVIRONMENT WOULD BE THE EDITOR. A DISCUSSION OF THIS MECHANISM IS INCLUDED BELOW IN THE APPENDIX.

3.6 CONTROL INSTRUCTIONS

SEVERAL OF THE MORE POWERFUL FEATURES DESCRIBED ABOVE (NOTABLY FUNCTIONS) REDUCE THE NUMBER OF PRIMITIVE REX INSTRUCTIONS NEEDED.

NOTE THAT THE CHOICE OF KEYWORDS IS FAIRLY ARBITRARY: AS NOTED EARLIER, THERE ARE ARGUABLE ALTERNATIVES TO MOST OF THEM.

IN THE FOLLOWING DIAGRAMS, SYMBOLS (WORDS) IN CAPITALS DENOTE KEYWORDS, OTHER WORDS (SUCH AS 'EXPRESSION') DENOTE A COLLECTION OF SYMBOLS AS DEFINED ABOVE. NOTE HOWEVER THAT THE KEYWORDS ARE NOT CASE DEPENDENT: THE SYMBOLS "IF", "If" AND "if" WOULD ALL INVOKE THE INSTRUCTION SHOWN BELOW AS "IF". NOTE ALSO THAT MANY OF THE DELIMITERS SHOWN MAY OFTEN BE OMITTED AS THEY WILL BE IMPLIED BY THE END OF A LINE. A 'THEN' IN THE CONTEXT OF A CLAUSE (IE AS THE FIRST AND ONLY SYMBOL) ACTS AS A SEMICOLON AS IS THEREFORE IGNORED.
THE CHARACTERS < AND > DELIMIT OPTIONAL PARTS OF THE INSTRUCTIONS.

3.6.1 ADDRESS

```
| ADDRESS <ENVIRONMENT <EXPRESSION>>;                          |
|                                                              |
| WHERE "ENVIRONMENT" IS A SINGLE SYMBOL OR STRING. IF THE SYMBOL |
| IS A VARIABLE THEN ITS VALUE IS USED.                        |
```

THIS INSTRUCTION IS USED TO EFFECT A TEMPORARY OR PERMANENT CHANGE TO THE DESTINATION OF COMMAND(S). THE CONCEPT OF ALTERNATIVE SUBCOMMAND ENVIRONMENTS IS DESCRIBED IN THE APPENDIX.

IF AN EXPRESSION IS GIVEN, IT WILL BE EVALUATED, AND THE RESULTING COMMAND STRING WILL BE ROUTED TO THE ENVIRONMENT WITH THE NAME GIVEN IN "ENVIRONMENT". AFTER EXECUTION OF THE COMMAND, THE ENVIRONMENT WILL BE SET BACK TO WHATEVER IT WAS BEFORE, THUS GIVING A TEMPORARY CHANGE OF DESTINATION FOR A SINGLE COMMAND.

IF NO EXPRESSION WAS SPECIFIED, THEN A LASTING CHANGE OF DESTINATION OCCURS: ALL FOLLOWING COMMANDS (EXPRESSIONS NOT PRECEDED BY A REX KEYWORD) WILL BE ROUTED TO THE COMMAND ENVIRONMENT WITH THE NAME GIVEN, UNTIL THE NEXT ADDRESS INSTRUCTION IS EXECUTED.

IF NO ARGUMENTS ARE GIVEN, COMMANDS WILL BE ROUTED BACK TO THE ENVIRONMENT THAT WAS SELECTED BEFORE PREVIOUS "ADDRESS <ENVIRONMENT>" INSTRUCTION, AND THE CURRENT ENVIRONMENT NAME IS SAVED. REPEATED EXECUTION OF "ADDRESS" WILL THEREFORE "TOGGLE" THE COMMAND DESTINATION BETWEEN TWO ENVIRONMENTS.

IF THE NULL STRING '' OR A BLANK STRING IS GIVEN AS THE ENVIRONMENT NAME, THEN THE HOST ENVIRONMENT (E.G. CMS) IS IMPLIED. THIS IS ALSO THE ENVIRONMENT SELECTED IF AN ADDRESS INSTRUCTION WITHOUT ARGUMENTS IS THE FIRST ADDRESS INSTRUCTION EXECUTED. IN THE CMS IMPLEMENTATION, THE NAME "CMS" ALSO IMPLIES THE HOST COMMAND ENVIRONMENT.

3.6.2 ARGS

```
ARGS     <TEMPLATE>;

WHERE "TEMPLATE" IS A LIST OF SYMBOLS SEPARATED BY BLANKS AND/OR
"TRIGGERS" (SPECIAL CHARACTERS OR STRINGS).
```

UNLESS A SUBROUTINE IS BEING EXECUTED (SEE THE CALL INSTRUCTION) THE INPUT PARAMETERS TO THE PROGRAM, PRECEDED BY THE NAME OF THE PROGRAM, WILL BE READ AS ONE STRING, AND THEN PARSED IN EXACTLY THE SAME WAY AS FOR THE PULL INSTRUCTION. (ARGS CAUSES TOKENS TO BE TRANSLATED TO UPPER CASE, USE THE PARSE ARGS INSTRUCTION IF THIS IS NOT DESIRED).

IF A SUBROUTINE IS BEING EXECUTED, THEN THE STRING USED WILL BE THE ARGUMENT STRING PASSED TO THE SUBROUTINE (PREFIXED WITH THE NAME OF THE SUBROUTINE).

THE ARGS (AND PARSE ARGS) INSTRUCTIONS MAY BE EXECUTED AS OFTEN AS DESIRED (TYPICALLY WITH DIFFERENT TEMPLATES) AND WILL ALWAYS PARSE THE SAME CURRENT INPUT STRING. THERE ARE NO RESTRICTIONS ON THE LENGTH OR CONTENT OF THE DATA PARSED EXCEPT THOSE IMPOSED BY THE CALLER. (E.G. FROM CMS COMMAND LEVEL, THE STRING WILL NOT EXCEED 130 CHARACTERS).

NOTE FOR EXEC USERS: UNLIKE EXEC AND EXEC 2, THE ARGUMENTS PASSED TO REX EXECS CAN ONLY BE USED AFTER EXECUTING EITHER THE ARGS OR PARSE ARGS COMMANDS. THEY ARE NOT IMMEDIATELY AVAILABLE IN PREDEFINED VARIABLES AS IN THE OTHER LANGUAGES.

3.6.3 CALL

```
| CALL NAME <EXPRESSION>;
```

CALL IS USED TO INVOKE AN INTERNAL SUBROUTINE. THE EXPRESSION (IF GIVEN) IS EVALUATED AND THEN PREFIXED WITH THE NAME OF THE SUBROUTINE: THE COMPLETE STRING IS THEN USED AS THE ARGUMENT STRING DURING EXECUTION OF THE SUBROUTINE (I.E. THE ARGS AND PARSE ARGS INSTRUCTION WILL ACCESS THIS STRING RATHER THAN THE ONE ACTIVE PREVIOUSLY).

THE CALL THEN CAUSES A BRANCH TO THE LABEL "NAME" USING EXACILY THE SAME MECHANISM AS THE SIGNAL INSTRUCTION (EXCEPT THAT PENDING DO-LOOPS ETC. ARE NOT DEACTIVATED). THE LINE NUMBER OF THE CALL INSTRUCTION IS AVAILABLE IN THE VARIABLE "SIGL" AS A DEBUG AID.

DURING EXECUTION OF THE SUBROUTINE, ALL VARIABLES PREVIOUSLY KNOWN ARE NORMALLY ACCESSIBLE. HOWEVER, THE PROCEDURE INSTRUCTION MAY USED TO SET UP A LOCAL VARIABLES ENVIRONMENT TO PROTECT THE SUBROUTINE AND CALLER FROM EACH OTHER.

EVENTUALLY THE SUBROUTINE SHOULD EXECUTE A RETURN INSTRUCTION, AND AT THAT POINT CONTROL WILL RETURN TO THE CLAUSE FOLLOWING THE ORIGINAL CALL, WITH THE VARIABLE "NAME" SET TO THE VALUE GIVEN ON THE RETURN STATEMENT (WHICH MAY BE NULL).

SUBROUTINES MAY BE NESTED TO ANY DEPTH.

NOTE THAT THE NAME GIVEN IN THE CALL INSTRUCTION MUST BE A VALID SYMBOL (NOT A LITERAL STRING) WHICH IS TREATED LITERALLY.

NOTE ALSO THAT EXECUTING A SIGNAL WHILE WITHIN A SUBROUTINE IS 'SAFE'. THAT IS, DO-LOOPS ETC. THAT WERE ACTIVE WHEN THE SUBROUTINE WAS CALLED ARE NOT DEACTIVATED. (BUT THOSE CURRENTLY ACTIVE ARE).

EXAMPLE:

```
/* RECURSIVE SUBROUTINE EXECUTION... */
ARGS NAME X
CALL FACTORIAL X
SAY X'  =' FACTORIAL
EXIT

FACTORIAL: PROCEDURE          /* CALCULATE FACTORIAL BY RECURSIVE */
  ARGS XX N                   /* INVOCATION...                    */
  IF N=0 THEN RETURN 1
  CALL FACTORIAL N-1
  RETURN FACTORIAL*N
```

3.6.4 DO

```
DO <REPETITOR> <CONDITIONAL>; <INSTRUCTION-LIST> END;

WHERE REPETITOR IS EITHER OF:
          NAME = EXPRI <TO EXPRT> <BY EXPRB>
          EXPRR
AND CONDITIONAL IS EITHER OF:
          WHILE EXPRW
          UNTIL EXPRU
AND INSTRUCTION-LIST IS: ANY SEQUENCE OF INSTRUCTIONS
```

NOTES:
THE TO AND BY PHRASES MAY BE IN REVERSE ORDER IF DESIRED.
EXPRR, EXPRI, EXPRB, AND EXPRT (IF PRESENT) MUST EVALUATE TO WHOLE NUMBERS.
EXPRW OR EXPRU (IF PRESENT) MUST EVALUATE TO '1' OR '0'.
EXPRB DEFAULTS TO '1', IF RELEVANT.
THE INSTRUCTION(S) IN INSTRUCTION-LIST MAY INCLUDE ANY OF THE MORE COMPLEX CONSTRUCTIONS SUCH AS IF, SELECT, OR THE DO INSTRUCTION ITSELF.
THE SUB-KEYWORDS TO BY WHILE AND UNTIL ARE RESERVED WITHIN A DO INSTRUCTION, IE THEY CANNOT NAME VARIABLES IN THE EXPRESSION(S).

THE DO INSTRUCTION IS USED TO GROUP STATEMENTS TOGETHER AND OPTIONALLY TO EXECUTE THEM REPETITIVELY.

IF NEITHER "REPETITOR" NOR "CONDITIONAL" IS GIVEN, THEN THE CONSTRUCT MERELY GROUPS A NUMBER OF INSTRUCTIONS TOGETHER: THESE ARE EXECUTED ONCE.

OTHERWISE THE INSTRUCTION-LIST IS EXECUTED ACCORDING TO THE REPETITOR PHRASE, OPTIONALLY MODIFIED BY THE CONDITIONAL PHRASE:

IF NO REPETITOR IS GIVEN (SO THERE IS ONLY A CONDITIONAL, SEE BELOW), THEN THE INSTRUCTION-LIST WILL NOMINALLY BE EXECUTED INDEFINITELY IE. UNTIL THE CONDITION IS SATISFIED.

IN THE SIMPLE FORM OF THE REPETITOR, THE EXPRESSION "EXPRR" IS EVALUATED IMMEDIATELY (RESULTING IN A NUMBER WHICH IS NOT NEGATIVE AND IS A WHOLE NUMBER), AND THE LOOP IS THEN EXECUTED THAT MANY TIMES (WHICH MAY BE 0). NOTE THAT, SIMILAR TO THE DIFFERENCE BETWEEN A COMMAND AND THE ASSIGNMENT STATEMENT, IF THE FIRST TOKEN OF "EXPRR" IS A SYMBOL AND THE SECOND TOKEN IS AN '=', THEN THE CONTROLLED FORM OF REPETITOR WILL BE EXPECTED:

THE CONTROLLED FORM SPECIFIES A CONTROL VARIABLE, "NAME", WHICH IS GIVEN AN INITIAL VALUE (THE RESULT OF "EXPRI"), AND WHICH IS THEN STEPPED (BY ADDING THE RESULT OF "EXPRB") EACH TIME THE INSTRUCTION-LIST IS EXECUTED, WHILE THE END CONDITION (THE RESULT OF "EXPRT") IS NOT EXCEEDED. IF "EXPRB" IS POSITIVE, THEN THE LOOP WILL BE TERMINATED WHEN

"NAME" IS GREATER THAN "EXPRT". IF NEGATIVE, THEN THE LOOP WILL BE TERMINATED WHEN "NAME" IS LESS THAN "EXPRT".
"EXPRI", "EXPRT", AND "EXPRB" MUST RESULT IN WHOLE NUMBERS, AND ARE EVALUATED ONCE ONLY, BEFORE THE LOOP BEGINS. THE DEFAULT VALUE FOR "EXPRB" IS 'I'. IF NO "EXPRT" IS GIVEN THEN THE LOOP WILL EXECUTE INDEFINITELY UNLESS SOME OTHER CONDITION TERMINATES IT.
NOTE THAT THE VARIABLE "NAME" CAN BE ALTERED WITHIN THE LOOP, AND THIS MAY AFFECT THE ITERATION OF THE LOOP. ALTERING THE VALUE OF THE CONTROL VARIABLE WHILE INSIDE A LOOP IS NOT NORMALLY CONSIDERED GOOD PROGRAMMING PRACTICE, THOUGH IT MAY BE USEFUL IN CERTAIN CIRCUMSTANCES.

ANY OF THE THREE FORMS OF REPETITOR (NONE, SIMPLE, OR CONTROLLED) MAY BE FOLLOWED BY A CONDITIONAL, WHICH MAY CAUSE TERMINATION OF THE LOOP. IF 'WHILE' OR 'UNTIL' IS SPECIFIED, THE EXPRESSION FOLLOWING IT IS EXECUTED EACH TIME AROUND THE LOOP (AND MUST EVALUATE TO EITHER '0' OR 'I'), AND THE INSTRUCTION-LIST WILL BE REPEATEDLY EXECUTED EITHER WHILE THE RESULT IS 'I', OR UNTIL THE RESULT IS '1'.
FOR A 'WHILE' LOOP, THE CONDITION IS EVALUATED AT THE TOP OF THE INSTRUCTION LIST, AND FOR AN 'UNTIL' LOOP THE CONDITION IS EVALUATED AT THE BOTTOM (AFTER THE VARIABLE "NAME" HAS BEEN STEPPED, IF APPROPRIATE).

NOTE THAT EXECUTION OF A CONTROLLED LOOP MAY BE MODIFIED BY USING THE LEAVE OR ITERATE STATEMENTS.

PROGRAMMER'S MODEL = HOW A TYPICAL DO LOOP IS EXECUTED:

FOR THE FOLLOWING DO:

```
DO NAME=EXPRI TO EXPRT BY EXPRB WHILE EXPRW
   ....
   INSTRUCTION-LIST
   ....
   END
```

REX WILL EXECUTE THE FOLLOWING:

```
   NAME=EXPRI
   £TEMPT=EXPRT     /* £VARIABLES ARE INTERNAL AND NOT ACCESSIBLE */
   £TEMPB=EXPRB
£LOOP:
   IF NAME > £TEMPT THEN LEAVE  /* LEAVE MEANS 'QUIT THE LOOP' */
   IF ¬EXPRW THEN LEAVE
      ....
      INSTRUCTION-LIST
      ....
   NAME=NAME + £TEMPB
   /* UNTIL EXPRESSION WOULD HAVE BEEN TESTED HERE, IF USED */
   TRANSFER CONTROL TO LABEL £LOOP
```

NOTE: THIS EXAMPLE IS FOR EXPRB >= 0. FOR NEGATIVE EXPRB, THE TEST AT THE START OF THE LOOP WOULD BE < RATHER THAN >

3.6.5 DROP

```
DROP <VARIABLE-LIST>;

WHERE VARIABLE-LIST IS A LIST OF SYMBOLS SEPARATED BY BLANKS.
```

EACH VARIABLE IN THE LIST WILL BE DROPPED FROM THE LIST OF KNOWN VARIABLES. IT IS NOT AN ERROR TO SPECIFY A NAME MORE THAN ONCE, OR TO 'DROP' A VARIABLE THAT IS NOT KNOWN. IN EITHER OF THESE CASES THE NAME IS IGNORED.

IF NO LIST IS SPECIFIED, THEN ALL KNOWN VARIABLES WILL BE DROPPED.

3.6.6 EXIT

```
EXIT <EXPRESSION>;
```

THE EXPRESSION IS CONVERTED TO A NUMBER, WHICH MUST BE A WHOLE NUMBER (IE THE DECIMAL PART, IF ANY, MUST BE 0), AND EXECUTION IS TERMINATED WITH THE NUMBER BEING USED TO SET THE RETURNCODE. IF NO EXPRESSION IS GIVEN, THE RETURNCODE WILL BE SET TO 0.

NOTE THAT "RUNNING OFF THE END" OF THE PROGRAM IS EQUIVALENT TO THE INSTRUCTION "EXIT 0".

3.6.7 IF

```
IF EXPRESSION THEN INSTRUCTION <ELSE <;> INSTRUCTION>;
                ;
```

THE EXPRESSION IS EVALUATED AND MUST RESULT IN '0' OR '1'. THE FIRST INSTRUCTION IS EXECUTED ONLY IF THE RESULT WAS '1'. IF AN ELSE WAS GIVEN, THEN THE INSTRUCTION AFTER THE ELSE IS EXECUTED ONLY IF THE RESULT WAS '0'.

THE SPECIAL KEYWORD 'THEN' MAY FOR CONVENIENCE BE USED TO TERMINATE THE IF CLAUSE, OR IS LEGAL AS A CLAUSE IN ITS OWN RIGHT. IN BOTH THESE CASES IT IS SYNONYMOUS WITH THE SEPARATOR CHARACTER (;) AND IS THEREFORE IGNORED AND SO MAY BE USED FOR IMPROVED READABILITY. HENCE A VARIABLE CALLED "THEN" CANNOT BE USED WITHIN THE EXPRESSION.

NOTE THAT AN "INSTRUCTION" INCLUDES ALL THE MORE COMPLEX CONSTRUCTIONS SUCH AS DO GROUPS, AS WELL AS THE SIMPLER ONES AND THE "IF" INSTRUCTION ITSELF.

3.6.8 ITERATE

```
%-------------------------------------------------------------@
|  ITERATE <SYMBOL>;                                          |
'-------------------------------------------------------------#
```

ITERATE ALTERS THE FLOW WITHIN A CONTROLLED DO LOOP (A LOOP WHICH IS REPEATED UNDER THE CONTROL OF A VARIABLE - EG: DO I=1 TO 10). EXECUTION OF THE INSTRUCTION LIST STOPS, AND CONTROL IS PASSED BACK UP TO THE DO CLAUSE JUST AS THOUGH THE END CLAUSE HAD BEEN ENCOUNTERED. THE CONTROL VARIABLE IS THEN STEPPED (ITERATED) AS NORMAL, AND THE INSTRUCTION LIST EXECUTED AGAIN UNLESS THE LOOP IS TERMINATED BY THE DO CLAUSE.

IF NO SYMBOL IS SPECIFIED, THEN ITERATE WILL STEP THE INNERMOST ACTIVE CONTROLLED LOOP. IF A SYMBOL IS SPECIFIED, THEN IT MUST BE THE NAME OF THE CONTROL VARIABLE OF A CURRENTLY ACTIVE LOOP (WHICH MAY BE THE INNERMOST), AND THIS IS THE LOOP THAT IS STEPPED. ANY ACTIVE LOOPS INSIDE THE ONE SELECTED FOR ITERATION ARE TERMINATED (AS THOUGH BY A LEAVE INSTRUCTION).

NOTES: A LOOP IS ACTIVE IF IT IS CURRENTLY BEING EXECUTED. IF A SUBROUTINE IS CALLED (OR A "REX" INSTRUCTION IS EXECUTED) DURING EXECUTION OF A LOOP, THEN THE LOOP BECOMES INACTIVE UNTIL THE SUBROUTINE HAS RETURNED OR THE WHOLE STRING HAS BEEN INTERPRETED.

IF MORE THAN ONE ACTIVE LOOP USES THE SAME CONTROL VARIABLE ( ) THEN THE INNERMOST WILL BE THE ONE SELECTED BY THE ITERATE.

3.6.9 LEAVE

```
%-------------------------------------------------------------------------@
| LEAVE <SYMBOL>;                                                         |
'-------------------------------------------------------------------------#
```

LEAVE CAUSES IMMEDIATE EXIT FROM ONE OR MORE CONTROLLED DO LOOPS (LOOPS WHICH ARE REPEATED UNDER THE CONTROL OF A VARIABLE - EG: DO I=1 TO 10). EXECUTION OF THE INSTRUCTION LIST IS TERMINATED, AND CONTROL IS PASSED TO THE INSTRUCTION FOLLOWING THE END CLAUSE, JUST AS THOUGH THE END CLAUSE HAD BEEN ENCOUNTERED AND THE TERMINATION CONDITION HAD BEEN MET NORMALLY, EXCEPT THAT ON EXIT THE CONTROL VARIABLE WILL CONTAIN THE VALUE IT HAD WHEN THE LEAVE INSTRUCTION WAS EXECUTED.

IF NO SYMBOL IS SPECIFIED, THEN LEAVE WILL TERMINATE THE INNERMOST ACTIVE CONTROLLED LOOP. IF A SYMBOL IS SPECIFIED, THEN IT MUST BE THE NAME OF THE CONTROL VARIABLE OF A CURRENTLY ACTIVE LOOP (WHICH MAY BE THE INNERMOST), AND THAT LOOP (AND ANY ACTIVE LOOPS INSIDE IT) IS THEN TERMINATED. CONTROL THEN PASSES TO THE CLAUSE FOLLOWING THE END THAT MATCHES THE DO CLAUSE OF THE SELECTED LOOP.

NOTES: A LOOP IS ACTIVE IF IT IS CURRENTLY BEING EXECUTED. IF A SUBROUTINE IS CALLED (OR A "REX" INSTRUCTION IS EXECUTED) DURING EXECUTION OF A LOOP, THEN THE LOOP BECOMES INACTIVE UNTIL THE SUBROUTINE HAS RETURNED OR THE WHOLE STRING HAS BEEN INTERPRETED.

IF MORE THAN ONE ACTIVE LOOP USES THE SAME CONTROL VARIABLE ( ) THEN THE INNERMOST WILL BE THE ONE SELECTED BY THE LEAVE.

3.6.10 NOP

```
%-------------------------------------------------------------------------@
| NOP;                                                                    |
'-------------------------------------------------------------------------#
```

NOP IS A DUMMY INSTRUCTION WHICH HAS NO EFFECT. IT CAN BE USEFUL AS THE TARGET OF AN ELSE, WHEN, OR THEN CLAUSE: EG:

```
SELECT
  WHEN A=B THEN NOP /* DO NOTHING */
  WHEN A>B THEN SAY 'A > B'
  OTHERWISE     SAY 'A < B'
END
```

3.6.11 PARSE

```
PARSE ARGS <TEMPLATE>;
PARSE PULL <TEMPLATE>;
PARSE VAR NAME <TEMPLATE>;

WHERE "TEMPLATE" IS A LIST OF SYMBOLS SEPARATED BY BLANKS AND/OR
"TRIGGERS" (SPECIAL CHARACTERS OR STRINGS).
```

THE PARSE INSTRUCTION IS USED TO PARSE DATA WITHOUT TRANSLATION OF WORDS TO UPPER CASE. THE TEMPLATE IS USED TO PARSE THE DATA FOLLOWING THE SAME RULES AS FOR THE PULL INSTRUCTION (Q.V.), EXCEPT FOR THE LACK OF TRANSLATION.

THE DATA USED FOR EACH VARIANT OF THE PARSE INSTRUCTION IS:

FOR PARSE ARGS: THE STRING PASSED TO THE PROGRAM OR SUBROUTINE AS AN INPUT PARAMETER LIST IS USED. (SEE UNDER THE ARGS INSTRUCTION FOR DETAILS)

FOR PARSE PULL: THE NEXT LINE FROM THE STACK (OR FAILING THAT FROM A CONSOLE READ) IS USED. (SEE ALSO THE PULL INSTRUCTION)

FOR PARSE VAR NAME: THE VALUE OF THE VARIABLE SPECIFIED BY "NAME" IS USED.

3.6.12 PROCEDURE

```
PROCEDURE;
```

THE PROCEDURE STATEMENT MAY BE USED WITHIN A SUBROUTINE TO PROTECT THE EXISTING VARIABLES BY MAKING THEM UNKNOWN TO FOLLOWING INSTRUCTIONS. ON EXECUTING A RETURN STATEMENT, THE ORIGINAL VARIABLES ARE RESTORED, AND ALL THOSE USED LOCALLY WITHIN THE SUBROUTINE ARE DROPPED.

A SUBROUTINE NEED NOT INCLUDE A PROCEDURE STATEMENT, IN WHICH CASE THE VARIABLES IT IS MANIPULATING ARE THOSE 'OWNED' BY THE CALLER.

ONLY ONE PROCEDURE STATEMENT IN EACH LEVEL OF SUBROUTINE CALL IS ALLOWED, ALL OTHERS (AND THOSE MET OUTSIDE SUBROUTINES) ARE IN ERROR.

THE PROCEDURE STATEMENT NEED NOT BE THE FIRST INSTRUCTION EXECUTED AFTER THE CALL: PREVIOUS INSTRUCTIONS WILL ACCESS THE CURRENT LEVEL OF VARIABLES.

3.6.13 PULL

```
PULL     <TEMPLATE>;

WHERE "TEMPLATE" IS A LIST OF SYMBOLS SEPARATED BY BLANKS AND/OR
"TRIGGERS" (SPECIAL CHARACTERS OR STRINGS).
```

THE CURRENT HEAD-OF-QUEUE WILL BE READ AS ONE STRING. IN THE CMS IMPLEMENTATION, THE CONSOLE INPUT "STACK" IS USED: IF THERE IS NOTHING IN THE STACK THEN A CONSOLE READ WILL OCCUR. IF NO TEMPLATE IS SPECIFIED, NO FURTHER ACTION IS TAKEN.

GENERAL RULES OF PARSING: (THE REST OF THIS SECTION APPLIES ALSO TO THE ARGS AND PARSE INSTRUCTIONS).

THE INPUT DATA STRING WILL BE PARSED ACCORDING TO THE RULES DEFINED BELOW, AND WORDS (GROUPS OF CHARACTERS DELIMITED BY BLANKS) ARE ASSIGNED TO THE TEMPLATE SYMBOLS IN SEQUENCE. FOR THE PULL AND ARGS INSTRUCTIONS THE WORDS ARE TRANSLATED TO UPPERCASE FIRST. (USE THE PARSE INSTRUCTION IF THIS IS NOT DESIRED).

UNLESS THERE ARE FEWER WORDS IN THE INPUT STRING THAN SYMBOLS IN THE TEMPLATE, THE FINAL SYMBOL WILL HAVE THE REMAINDER OF THE INPUT LINE ASSIGNED TO IT AS A SINGLE UNEDITED STRING. THUS IN THE LIMITING CASE OF THERE ONLY BEING ONE SYMBOL SPECIFIED, IT WILL BE ASSIGNED THE ENTIRE INPUT STRING.

IF YOU ARE INTERESTED IN FEWER WORDS THAN EXIST IN THE STRING, THEN YOU WILL WANT TO ADD A DUMMY VARIABLE TO THE END OF THE TEMPLATE (TO AVOID THE FINAL VARIABLE BEING ASSIGNED MULTIPLE WORDS).

ALL SYMBOLS IN THE TEMPLATE WILL BE GIVEN A NEW VALUE, AND ARE THEREBY CONTEXTUALLY DECLARED AS VARIABLES. (IF THERE ARE MORE SYMBOLS THAN WORDS IN THE DATA, THEN THE EXTRA VARIABLES WILL BE ASSIGNED THE NULL STRING.)

THE VALUES ASSIGNED TO EACH VARIABLE MAY BE INSPECTED DURING PROGRAM DEBUG BY USING THE "TRACE RESULTS" INSTRUCTION (Q.V.).

SIMPLE PARSING (NO TRIGGERS IN THE TEMPLATE):

FOR EACH VARIABLE EXCEPT THE FINAL ONE IN THE TEMPLATE, THE INPUT LINE IS SCANNED FOR "WORDS" (IE, CONTIGUOUS NON-BLANK CHARACTERS). EACH WORD (TRANSLATED TO UPPER CASE FOR PULL AND ARGS, OR LEFT UNTRANSLATED FOR PARSE) IS THEN ASSIGNED TO THE NEXT VARIABLE SPECIFIED IN THE LIST. THE FINAL VARIABLE IN THE LIST WILL HAVE ANY REMAINDER OF THE INPUT LINE ASSIGNED TO IT WITHOUT ANY EDITING OR TRANSLATION.

LEADING AND TRAILING BLANKS ARE REMOVED FROM ANY WORD BEING ASSIGNED TO A VARIABLE, BUT ARE NOT REMOVED FROM THE STRING ASSIGNED TO THE FINAL VARIABLE. THEREFORE IF THERE IS ONLY ONE SYMBOL IN THE TEMPLATE, IT WILL BE ASSIGNED THE ENTIRE LINE COMPLETE WITH LEADING AND TRAILING BLANKS. IF HOWEVER THERE IS MORE THAN ONE SYMBOL, THEN THE FINAL VARIABLE WILL BE ASSIGNED A VALUE WHICH MAY HAVE TRAILING BLANKS BUT WHICH WILL NEVER HAVE LEADING BLANKS.

EXAMPLE:

```
DATA ON THE STACK:    ' FRED  PLEASE  TALK TO ME  '
INSTRUCTION:          "PULL NAME REST"
RESULTS IN:           "NAME" HAS THE VALUE 'FRED'
                      "REST" HAS THE VALUE 'PLEASE  TALK TO ME  '

THEN THE INSTRUCTION: "PARSE VAR REST WRD1 WRD2 REST"

RESULTS IN:           "WRD1" HAS THE VALUE 'PLEASE'
                      "WRD2" HAS THE VALUE 'TALK'
                      "REST" HAS THE VALUE 'TO ME  '
```

PARSING WITH TRIGGERS:

REX ALLOWS SIMPLE PATTERN MATCHING USING TRIGGERS (ANY SPECIAL CHARACTER OR STRING) IN THE TEMPLATE.

TRIGGERS IN THE TEMPLATE CAUSE THE LIST OF SYMBOLS TO BE SEPARATED INTO GROUPS, EACH OF WHICH IS MATCHED AGAINST A STRING OF DATA CHARACTERS WHICH IS THEN PARSED IN THE SAME WAY AS DESCRIBED ABOVE FOR 'SIMPLE PARSING' (I.E. IF THERE IS MORE THAN ONE SYMBOL, ALL EXCEPT THE LAST ARE ASSIGNED INDIVIDUAL WORDS, AND THE FINAL ONE WILL GET THE DATA REMAINING).

THE PRESENCE OF A TRIGGER IN THE TEMPLATE WILL CAUSE THE DATA STRING TO BE FORWARD SCANNED FOR THAT TRIGGER. THE DATA UP TO THE TRIGGER IS THEN USED FOR PARSING AGAINST THE CURRENT GROUP OF TEMPLATE SYMBOLS, OR IF THE TRIGGER IS NOT FOUND, ALL THE REMAINING DATA IS USED.

ONCE DATA IS PARSED AND ASSIGNED TO A GROUP OF SYMBOLS, IT (AND THE TRIGGER CHARACTER(S)) IS SKIPPED OVER AND MATCHED AGAINST THE NEXT GROUP OF SYMBOLS IN THE TEMPLATE (UP TO THE NEXT TRIGGER OR THE END OF THE TEMPLATE). DATA IS THEREFORE NEVER SCANNED MORE THAN ONCE DURING PARSING.

EXAMPLE 1 (SINGLE SPECIAL CHARACTER TRIGGER):

```
DATA ON THE STACK:    'FRED WAS HERE (I THINK)'
INSTRUCTION:          "PULL NAME REST ( MORE"
RESULTS IN:           "NAME" HAS THE VALUE 'FRED'
                      "REST" HAS THE VALUE 'WAS HERE'
                      "MORE" HAS THE VALUE 'I THINK)'
```

EXAMPLE 2 (ONE STRING AND ONE SPECIAL CHARACTER TRIGGER):

```
THE ARGUMENT STRING:  'SEND FRED TO THERE (PLEASE'
INSTRUCTION:          "ARGS NAME1 NAME2 ' TO ' PLACE ( MORE"
RESULTS IN:           "NAME1" HAS THE VALUE 'FRED'
                      "NAME2" HAS THE VALUE ''
                      "PLACE" HAS THE VALUE 'THERE'
                      "MORE"  HAS THE VALUE 'PLEASE'
```

3.6.14 PUSH

```
| PUSH <EXPRESSION>;
```

THE STRING RESULTING FROM EXPRESSION WILL BE STACKED LIFO (LAST IN, FIRST OUT). IF NO EXPRESSION IS SPECIFIED, THE NULL STRING IS STACKED. IN THE CMS IMPLEMENTATION, THE CONSOLE INPUT STACK IS USED.

3.6.15 QUEUE

```
| QUEUE <EXPRESSION>;
```

THE STRING RESULTING FROM EXPRESSION WILL BE "STACKED" FIFO (FIRST IN, FIRST OUT). IF NO EXPRESSION IS SPECIFIED, THE NULL STRING IS QUEUED. IN THE CMS IMPLEMENTATION, THE CONSOLE INPUT STACK IS USED.

3.6.16 RETURN

```
| RETURN <EXPRESSION>;
```

IF A SUBROUTINE IS BEING EXECUTED (SEE THE CALL INSTRUCTION) THEN THE EXPRESSION IS EVALUATED, AND THE VARIABLE SPECIFIED BY THE SUBROUTINE NAME IS SET TO THE RESULT. (I.E, IF THE SUBROUTINE WAS CALLED "FRED", THEN THE VARIABLE "FRED" WOULD BE SET). IF NO EXPRESSION IS SPECIFIED, THEN THE VARIABLE IS SET TO NULL.

ONCE THE VARIABLE HAS BEEN SET, CONTROL RETURNS TO THE CLAUSE FOLLOWING THE CALL INSTRUCTION, AND THE VARIABLE "SIGL" IS SET TO THE LINE NUMBER OF THE RETURN INSTRUCTION JUST EXECUTED.

IF A PROCEDURE STATEMENT WAS EXECUTED WITHIN THE SUBROUTINE, THEN THE LOCAL VARIABLES ARE DROPPED (AND THE OLD ONES RESTORED) AFTER THE EXPRESSION EVALUATION AND BEFORE THE RESULT IS ASSIGNED TO THE RETURN VARIABLE.

IF NO SUBROUTINE IS BEING EXECUTED, THEN RETURN WILL CAUSE CONTROL TO LEAVE THE PROGRAM: THE RESULT OF THE EXPRESSION IS PUSHED ONTO THE STACK, AND EXECUTION IS TERMINATED WITH RETURNCODE SET TO ZERO. IF NO EXPRESSION IS GIVEN, THE NULL STRING WILL BE PUSHED ONTO THE STACK.
NOTE: IF THE REX PROGRAM HAS BEEN CALLED (BY REX) AS A FUNCTION, THEN THE RETURN STATEMENT WILL AUTOMATICALLY USE THE DIRECT METHOD OF PASSING BACK DATA DESCRIBED ELSEWHERE. IN THIS CASE, THERE IS NO RESTRICTION ON THE LENGTH OR CONTENT OF THE DATA; AND NOTHING IS PLACED ON THE STACK.

3.6.17 REX

```
%-------------------------------------------------------------------@
|  REX <EXPRESSION>;                                                |
'-------------------------------------------------------------------#
```

THE EXPRESSION IS EVALUATED, AND WILL THEN BE EXECUTED (INTERPRETED) JUST AS THOUGH THE RESULTING STRING WERE A LINE INSERTED INTO THE INPUT FILE BRACKETED BY A DO AND AN END. ANY INSTRUCTIONS (INCLUDING "REX" INSTRUCTIONS) ARE ALLOWED, BUT NOTE THAT CONSTRUCTIONS SUCH AS DO ... END, SELECT ... END MUST BE COMPLETE.

A SEMICOLON IS IMPLIED AT THE END OF THE EXPRESSION DURING EXECUTION, AS A SERVICE TO THE USER.

IF NO EXPRESSION IS SPECIFIED, THE VERSION NUMBER AND DATE OF THE REX MODULE WILL BE DISPLAYED.

NOTE: LABELS WITHIN THE INTERPRETED STRING ARE NOT PERSISTENT AND ARE THEREFORE IGNORED. HENCE EXECUTING A SIGNAL INSTRUCTION FROM WITHIN AN INTERPRETED STRING WILL CAUSE IMMEDIATE EXIT FROM THAT STRING BEFORE THE LABEL SEARCH BEGINS.

3.6.18 SAY

```
%-------------------------------------------------------------------@
|  SAY <EXPRESSION>;                                                |
'-------------------------------------------------------------------#
```

THE STRING RESULTING FROM EXPRESSION IS DISPLAYED (OR SPOKEN, OR TYPED, ETC.) TO THE USER VIA WHATEVER CHANNEL IS IMPLEMENTED. ANY LENGTH STRING IS PERMITTED.

NOTE: IN THE CMS IMPLEMENTATION, THE DATA WILL BE FORMATTED WITH 80 CHARACTERS PER LINE. CONTINUATION LINES WILL THEN BE UP TO 79 CHARACTERS LONG; PRECEDED BY A SINGLE BYTE X'FF'. THIS ALLOWS PROGRAMS WHICH CAPTURE AND PROCESS THE OUTPUT FROM A REX PROGRAM TO RE-CREATE THE ORIGINAL FORMAT OF THE DATA WITHOUT LOSS OF INFORMATION. LINES ARE TYPED ON A TYPEWRITER TERMINAL, OR 'DISPLAYED' ON A VDU.

3.6.19 SELECT

```
SELECT; WHEN-LIST <OTHERWISE <;> <INSTRUCTION-LIST>> END;

WHERE WHEN-LIST      IS: ONE OR MORE WHEN-CONSTRUCT
AND   WHEN-CONSTRUCT IS: WHEN EXPRESSION THEN INSTRUCTION
                                          ;
AND INSTRUCTION-LIST IS ANY SEQUENCE OF INSTRUCTIONS
```

EACH EXPRESSION FOLLOWING A WHEN IS EVALUATED IN TURN AND MUST RESULT IN '0' OR '1'. IF THE RESULT IS '1', THE FOLLOWING INSTRUCTION (WHICH MAY BE A COMPLEX INSTRUCTION SUCH AS IF, DO, OR SELECT) IS EXECUTED AND CONTROL WILL THEN PASS TO THE END. OTHERWISE CONTROL WILL PASS TO THE NEXT WHEN STATEMENT.

IF NONE OF THE WHEN EXPRESSIONS SUCCEED, CONTROL WILL PASS TO THE INSTRUCTION-LIST (IF ANY) FOLLOWING OTHERWISE. IN THIS SITUATION, THE ABSENCE OF AN OTHERWISE WILL CAUSE AN ERROR.

THE SPECIAL KEYWORD 'THEN' MAY FOR CONVENIENCE BE USED TO TERMINATE A WHEN CLAUSE, OR IS LEGAL AS A CLAUSE IN ITS OWN RIGHT. IN BOTH THESE CASES IT IS SYNONYMOUS WITH THE SEPARATOR CHARACTER (;) AND IS THEREFORE IGNORED AND SO MAY BE USED FOR IMPROVED READABILITY. A VARIABLE CALLED "THEN" CANNOT THEREFORE BE USED WITHIN THE EXPRESSION IN A WHEN-CONSTRUCT.

3.6.20 SIGNAL AND LABELS

```
SIGNAL EXPRESSION;
SIGNAL ON  EXPRESSION;
SIGNAL OFF EXPRESSION;
```

THE SIGNAL INSTRUCTION CAUSES AN ABNORMAL CHANGE IN THE FLOW OF CONTROL, OR (IF ON OR OFF IS SPECIFIED) CONTROLS EXCEPTION HANDLING.

IN THE CASE OF NEITHER ON NOR OFF BEING SPECIFIED:
THE EXPRESSION IS EVALUATED, AND TAKEN TO BE THE NAME OF A LABEL. ALL PENDING LOOPS, "REX" INSTRUCTIONS, ETC. ARE THEN TERMINATED (I.E. THEY CANNOT BE REACTIVATED). IF THE LABEL IS KNOWN TO THE INTERPRETER (DUE TO ITS BEING SIGNALED OR CALLED BEFORE) THEN CONTROL WILL IMMEDIATELY BE PASSED TO THE CLAUSE FOLLOWING THE LABEL. OTHERWISE, A SEARCH IS MADE FOR THE LABEL: THE SEARCH PROGRESSES FROM THE TOP OF THE DATA THROUGH THE END OF THE DATA, WITH ALL THE DATA BEING SCANNED (REGARDLESS OF ANY FLOW CONTROL STATEMENTS). NOTE THAT SINCE REX 'REMEMBERS' LABELS, CONTROL WILL ALWAYS PASS TO THE FIRST ONE IN THE DATA IF DUPLICATES ARE PRESENT. (I.E, DUPLICATE LABELS ARE IGNORED).

IN THE CASE OF ON OR OFF BEING SPECIFIED:
THE EXPRESSION MUST EVALUATE TO ONE OF THE THREE BUILT-IN CONDITIONS:

ERROR - RAISED IF ANY HOST COMMAND RETURNS A NON-ZERO RETURNCODE.
EXIT - RAISED IF ANY EXIT (BUT NOT RETURN) INSTRUCTION IS EXECUTED.
SYNTAX - RAISED IF ANY INTERPRETATION ERROR IS DETECTED.

IF ON IS SPECIFIED, THE GIVEN CONDITION IS ENABLED; AND IF OFF IS GIVEN, THE CONDITION IS DISABLED. THE INITIAL SETTING OF ALL CONDITIONS IS OFF.

WHEN A CONDITION IS ENABLED AND THE SPECIFIED EVENT OCCURS, ALL CURRENT EXECUTION WILL CEASE, AND A 'SIGNAL XXX' (WHERE XXX IS ERROR, EXIT, OR SYNTAX) IS EXECUTED AUTOMATICALLY. WHEN THE EVENT OCCURS, THE CONDITION WILL BE DISABLED IMMEDIATELY (BUT MAY BE RE-ENABLED IF DESIRED). THEREFORE, IF THE LABEL IS NOT FOUND, A NORMAL SYNTAX ERROR EXIT WILL BE TAKEN.

NOTE THAT IF A SIGNAL INSTRUCTION OR CONDITION IS ISSUED AS A RESULT OF A REX INSTRUCTION, THE REMAINDER OF THE STRING(S) BEING INTERPRETED WILL NOT BE SEARCHED FOR THE GIVEN LABEL. IN EFFECT, LABELS WITHIN INTERPRETED STRINGS ARE IGNORED.

LABELS ARE SINGLE SYMBOLS OF UP TO 150 CHARACTERS, FOLLOWED BY A COLON. THE COLON ACTS AS A CLAUSE SEPARATOR, AND SO A LABEL IS A CLAUSE IN ITS OWN RIGHT AND MULTIPLE LABELS MAY THEREFORE PRECEDE AN EXECUTABLE CLAUSE.

FOLLOWING THE EXECUTION OF ANY JUMP DUE TO A SIGNAL, THE LINE NUMBER OF THE STATEMENT CAUSING THE JUMP IS STORED IN THE BUILT-IN VARIABLE 'SIGL'. THIS IS ESPECIALLY USEFUL FOR 'SIGNAL ON SYNTAX' (SEE ABOVE) WHEN THE NUMBER OF THE LINE IN ERROR CAN BE USED, FOR EXAMPLE, TO CONTROL AN EDITOR.

3.6.21 TRACE

```
%-------------------------------------------------------------------@
|  TRACE <EXPRESSION>;                                              |
'-------------------------------------------------------------------#
```

TRACE WILL EVALUATE THE EXPRESSION, AND TAKE ACTION ACCORDING TO THE FIRST CHARACTER OF THE RESULT:

'E' (E.G: "ERROR")	- HOST COMMANDS RESULTING IN NON-0 RETURNCODE ARE TRACED (AFTER EXECUTION)
'C' (E.G: "COMMANDS")	- ALL HOST COMMANDS ARE TRACED BEFORE EXECUTION; AND NON-ZERO RC IS SHOWN.
'A' (E.G: "ALL")	- ALL CLAUSES ARE TRACED BEFORE EXECUTION
'R' (E.G: "RESULTS")	- ALL CLAUSES ARE TRACED BEFORE EXECUTION TOGETHER WITH THE FINAL RESULT OF ANY EXPRESSION EVALUATED. VALUES ASSIGNED DURING PULL, ARGS AND PARSE INSTRUCTIONS ARE ALSO DISPLAYED.
'I' (E.G: "INTS")	- AS 'R' EXCEPT THAT ALL TERMS AND INTERMEDIATE RESULTS DURING EXPRESSION EVALUATION (AND SUBSTITUTED NAMES) ARE ALSO TRACED.
'S' (E.G: "SCAN")	- ALL REMAINING CLAUSES IN THE DATA WILL BE TRACED WITHOUT BEING EXECUTED. BASIC CHECKING (FOR MISSING END'S ETC) IS CARRIED OUT, AND THE TRACE IS FORMATTED AS USUAL. THIS ONLY HAS EFFECT IF THE "TRACE SCAN" CLAUSE IS NOT ITSELF NESTED IN ANY OTHER INSTRUCTION.
'O' (E.G: "OFF")	- NOTHING IS TRACED

(OTHER FIRST CHARACTERS WILL CAUSE AN ERROR)

ANY OF THESE OPTIONS MAY BE PREFIXED WITH THE CHARACTER ' ', WHICH HAS THE EFFECT OF INHIBITING COMMAND EXECUTION UNTIL THE NEXT TRACE COMMAND IS EXECUTED. THIS MAY BE USED FOR DEBUGGING POTENTIALLY DESTRUCTIVE PROGRAMS. FOR EXAMPLE, "TRACE COMMANDS" WILL CAUSE COMMANDS TO BE TRACED BUT NOT EXECUTED. IN THIS CASE, RC IS SET TO '0' AS EACH COMMAND IS BYPASSED.

COMMANDS TRACED BEFORE EXECUTION ALWAYS HAVE THE VALUE OF THE COMMAND TRACED AS WELL AS THE CLAUSE CAUSING IT.

EACH CLAUSE TRACED WILL BE DISPLAYED WITH AUTOMATIC FORMATTING (INDENTATION) ACCORDING TO ITS LOGICAL DEPTH OF NESTING ETC. RESULTS (IF REQUESTED) ARE INDENTED AN EXTRA TWO SPACES.

THE FIRST CLAUSE TRACED ON ANY LINE WILL HAVE ITS LINE NUMBER PREFIXED.

EXCEPT FOR "TRACE SCAN", CLAUSES ARE IDENTIFIED DURING TRACING BY A PRECEDING '*-*' TAG, AND RESULTS ARE IDENTIFIED BY A TAG OF '>>>'.

FOLLOWING A SYNTAX ERROR, THE STATEMENT IN ERROR WILL ALWAYS BE TRACED, AS WILL ANY CALL OR REX STATEMENTS ACTIVE AT THE TIME OF THE ERROR. THESE TRACEBACK LINES ARE IDENTIFIED BY THE SPECIAL TAG '+++'.

3.7 FUNCTIONS AND BUILT-IN VARIABLES

SINCE REX CAN INCLUDE EXTERNAL FUNCTION CALLS DIRECTLY IN EXPRESSIONS, NO BUILT-IN FUNCTIONS HAVE BEEN IMPLEMENTED.

A REFERENCE TO A CMS MODULE, NUCX, OR ANOTHER REX EXEC MAY BE INCLUDED IN AN EXPRESSION USING THE CONVENTIONAL PARENTHESES NOTATION. NOTE THAT THE NAME OF THE FUNCTION MUST BE ADJACENT TO THE '(', WITH NO BLANK IN BETWEEN, OR THERE WILL BE A "BLANK OPERATOR" ASSUMED AT THIS POINT AND THE CONSTRUCT WILL NOT BE RECOGNISED AS A FUNCTION CALL.

EACH ARGUMENT EXPRESSION (SEPARATED BY COMMAS) IS EVALUATED, AND THE FUNCTION IS EXPECTED TO RETURN A RESULT IN AN ALLOCATED STORAGE BLOCK (SEE ELSEWHERE FOR DETAILS). IF THE FUNCTION DETECTS AN ERROR, IT SHOULD RETURN A NON-ZERO RETURNCODE (THE ACTUAL VALUE OF THIS IS IGNORED BY REX).

FOR EXAMPLE, THE FUNCTION SUBSTR IS INCLUDED IN THE REXFNS PACK (SEE BELOW) AND COULD BE USED AS:

```
"A=SUBSTR(B,2,7);"
```

NOTE THAT THE SYMBOL BEFORE THE '(' MAY BE A VARIABLE, SO THE SEQUENCE: "S=SUBSTR; A=S(B,2,7);" WOULD HAVE THE SAME RESULT AS THE EARLIER LINE. A LITERAL STRING, TOO, MAY BE USED TO HOLD THE FUNCTION NAME AND SO INVOKE THE FUNCTION.

A FUNCTION MAY HAVE A VARIABLE NUMBER OF ARGUMENTS: ONLY THOSE REQUIRED NEED BE SPECIFIED. SUBSTR(ABCDEF,4) WOULD RETURN 'DEF' FOR EXAMPLE.

OTHER REX EXEC'S MAY BE CALLED AS FUNCTIONS. E.G:

```
SAY 'EXEC SWAP'(ABC DEF);  /* SWAP SWAPS THE WORDS */
```

ONLY ONE PARAMETER STRING (UP TO 240 CHARACTERS) MAY BE INCLUDED BETWEEN THE PARENTHESES IN THIS CASE. THE RETURN INSTRUCTION THEN USES THE NORMAL REX FUNCTION INTERFACE (INSTEAD OF THE STACK) AND SO THERE IS NO RESTRICTION ON THE CONTENT OR LENGTH OF THE RETURNED CHARACTER STRING.

AN EXEC CALLED AS A FUNCTION SHOULD NORMALLY TERMINATE WITH A RETURN STATEMENT, AS USE OF THE EXIT STATEMENT WILL CAUSE AN ERROR WHEN CONTROL IS RECEIVED BACK BY THE CALLER.

BUILT-IN VARIABLES:

THE VARIABLE "RC" IS EQUIVALENT TO A VARIABLE CONTAINING THE RETURNCODE FROM THE LAST EXECUTED HOST COMMAND (OR SUBCOMMAND). FOLLOWING A SIGNAL EVENT (EXIT, SYNTAX, OR ERROR) IT CONTAINS THE CODE APPROPRIATE TO THE EVENT, IE THE VALUE ON THE EXIT STATEMENT, THE SYNTAX ERROR NUMBER (1-99), OR THE COMMAND RETURNCODE.

THE VARIABLE "SIGL" CONTAINS THE LINE NUMBER OF THE LAST STATEMENT THAT CAUSED A 'SIGNAL' JUMP (IE, A SIGNAL, CALL/RETURN, OR ERROR CONDITION).

NEITHER OF THESE VARIABLES HAVE AN INITIAL VALUE. THEY MAY BE ALTERED BY THE USER, JUST LIKE ANY OTHER VARIABLE, AND THEY MAY BE ACCESSED VIA THE DIRECT INTERFACE TO REX VARIABLES.

3.8 NUMERICS AND ARITHMETIC OPERATIONS

MOST ARITHMETIC OPERATIONS IN REX WILL INVOLVE INTEGERS (WHOLE NUMBERS WITH NO DECIMAL PART). OPERATIONS ON INTEGERS ARE THEREFORE EXPECTED TO RESULT IN INTEGERS, AND IN PARTICULAR, THE RESULT OF A DIVIDE IS TRUNCATED. (E.G. "3/2" WILL RESULT IN '1').

HOWEVER DECIMAL NUMBERS ARE OFTEN USEFUL FOR CALCULATIONS, AND SO REX ALLOWS UP TO NINE DIGITS AFTER A DECIMAL POINT.

THE PRECISION OF A RESULT IS DETERMINED BY THE PRECISION OF THE TERMS INVOLVED IN THE COMPUTATION: THE RESULT IS GIVEN TO THE SAME PRECISION AS THE MORE PRECISE OF THE TWO TERMS. THIS IS BEST ILLUSTRATED BY SOME EXAMPLES:

```
"8/3"      ==>  '2'     (INTEGER ARITHMETIC)
"8/3.0"    ==>  '2.7'   (NOTE ROUND UP ON DIVIDE)
"8/3.00"   ==>  '2.67'
ETC.
```

THIS MECHANISM ALLOWS REX TO EVALUATE INTEGER ARITHMETIC VERY EFFICIENTLY, AND AT THE SAME TIME PROVIDE A USEFUL DECIMAL ARITHMETIC FACILITY.

(NOTE: IN THE CURRENT IMPLEMENTATION A DIVISION BY A NUMBER WITH MORE THAN 9 SIGNIFICANT DIGITS MAY RESULT IN FIGURE WHICH IS ONLY ACCURATE TO NINE FIGURES. THIS RESTRICTION WILL BE LIFTED AS SOON AS POSSIBLE. ALL OTHER RESULTS ARE 'EXACT'.)

3.9 VARIABLES AND COMPOUND SYMBOLS (ARRAY HANDLING)

A SYMBOL WHICH HAS BEEN GIVEN A NEW VALUE (BY AN ASSIGNMENT, OR A PULL, ARGS, OR PARSE STATEMENT) IS CALLED A VARIABLE. THE "VALUE" OF A SYMBOL IS EITHER THE STRING ASSIGNED TO IT (IF A VARIABLE) OR OTHERWISE IS ITS DERIVED NAME. THE DERIVED NAME OF A SIMPLE SYMBOL IS THE UPPER CASE FORM OF THE SYMBOL, AS DESCRIBED EARLIER.

A SYMBOL WHICH DOES NOT START WITH A DIGIT (0-9) OR A PERIOD, YET INCLUDES AT LEAST ONE PERIOD, IS COMPOUND: THIS MEANS THAT ITS NAME MAY INCLUDE THE VALUE OF ONE OR MORE SYMBOLS.

COMPOUND SYMBOLS MAY THEREFORE BE USED TO SET UP ARRAYS AND LISTS OF VARIABLES, AND OFFER GREAT SCOPE FOR THE CREATIVE PROGRAMMER.

THE DERIVED NAME OF A COMPOUND VARIABLE OF THE FORM:

S0.S1.S2. --- .SN

IS THEN GIVEN BY:

D0.V1.V2. --- .VN

WHERE D0 IS THE UPPER CASE FORM OF THE SYMBOL S0, AND V1 TO VN ARE THE VALUES OF THE SIMPLE SYMBOLS S1 TO SN. ANY OF THE SYMBOLS S1-SN AND VALUES V1-VN MAY BE NULL.

SOME EXAMPLES FOLLOW IN THE FORM OF A SMALL EXTRACT FROM A FICTITIOUS REX EXEC:

```
A=3             /* ASSIGNS '3' TO THE VARIABLE WITH NAME 'A'       */
B=4                          /*   '4'       TO VAR NAMED 'B'       */
C='FRED'                     /*   'FRED'    TO VAR NAMED 'C'       */
A.B='FRED'                   /*   'FRED'    TO VAR NAMED 'A.4'     */
A.FRED=5                     /*   '5'       TO VAR NAMED 'A.FRED'  */
A.C='BILL'                   /*   'BILL'    TO VAR NAMED 'A.FRED'  */
C.C=A.FRED                   /*   '5'       TO VAR NAMED 'C.FRED'  */
X.A.B='ANNIE'                /*   'ANNIE'   TO VAR NAMED 'X.3.4'   */

SAY  A  B  C  A.A  A.B  A.C  C.A  A.FRED X.A.4 ?.A

/* WILL TYPE THE STRING: '3 4 FRED A.3 FRED BILL C.3 5 ANNIE ?.3' */
```

3.10 "RESERVED" KEYWORDS AND LANGUAGE EXTENDABILITY

THE FREE SYNTAX OF REX IMPLIES THAT SOME SYMBOLS ARE RESERVED FOR USE BY THE INTERPRETER IN CERTAIN CONTEXTS.

ONLY NON-COMPOUND SYMBOLS THAT ARE THE FIRST IN A CLAUSE AND THAT ARE NOT FOLLOWED BY AN "=" ARE CHECKED TO SEE IF THEY ARE KEYWORDS: THE SYMBOLS MAY BE FREELY USED ELSEWHERE IN CLAUSES WITHOUT BEING TAKEN TO BE KEYWORDS.

WITHIN PARTICULAR INSTRUCTIONS, SOME SYMBOLS MAY BE RESERVED TO SEPARATE THE PARTS OF THE INSTRUCTION: FOR EXAMPLE THE WHILE ETC. IN A DO INSTRUCTION, OR THE THEN (WHICH ACTS AS A CLAUSE TERMINATOR IN THIS CASE) IN AN IF OR WHEN CLAUSE.

THE KEYWORDS CAN THEREFORE ONLY ADVERSELY AFFECT THE USER IF IT IS DESIRED TO EXECUTE A HOST COMMAND OR SUBCOMMAND WITH THE SAME NAME (E.G. "QUEUE") AS A REX KEYWORD.

THIS IS POTENTIALLY A PROBLEM FOR ANY PROGRAMMER WHOSE REX PROGRAMS MIGHT BE USED FOR SOME TIME AND IN CIRCUMSTANCES OUTSIDE HIS OR HER CONTROL, AND WHO WISHES TO MAKE THE PROGRAMS ABSOLUTELY 'WATERTIGHT'.

IN THIS CASE A REX PROGRAM MAY BE WRITTEN WITH (AT LEAST) THE FIRST WORDS IN COMMAND LINES ENCLOSED IN QUOTES. E.G: 'ERASE' FN FT FM. THIS ALSO HAS AN ADVANTAGE IN THAT IT IS MORE EFFICIENT.

AN ALTERNATIVE STRATEGY IS TO PRECEDE SUCH COMMAND STRINGS WITH TWO ADJACENT QUOTES, WHICH WILL HAVE THE EFFECT OF CONCATENATING THE NULL STRING ON TO THE FRONT. E.G: ''ERASE FN FT FM. A THIRD BUT MORE UGLY OPTION IS TO ENCLOSE THE ENTIRE EXPRESSION (OR THE FIRST SYMBOL) IN PARENTHESES. E.G: (ERASE FN FT FM).

IMPORTANTLY, THE CHOICE OF STRATEGY (IF IT IS TO BE DONE AT ALL) IS A PERSONAL ONE BY THE PROGRAMMER, AND IS NOT IMPOSED BY THE REX LANGUAGE.

THE POSSIBILITY OF IDENTIFYING ALL REX KEYWORDS BY STARTING THEM WITH A UNIQUE CHARACTER (E.G. ".") WAS MOST SERIOUSLY CONSIDERED, HOWEVER THIS:

- DOES NOT SOLVE THE PROBLEM SHOULD ONE IN FUTURE BE ALLOWED COMMANDS STARTING WITH THAT SAME LETTER

- DESTROYS THE NATURAL LOOK OF THE LANGUAGE WHICH WAS ONE OF THE PRIME REASONS FOR ITS INCEPTION.

IN ADDITION TO THIS, IT WAS FELT THAT THE PROBLEM IS MUCH LESS SEVERE THAN THAT OF CHANGES TO THE HOST COMMANDS INVOKED BY THE PROGRAM: THESE ARE OFTEN FAR LESS CONTROLLED AND MAY EVEN HAVE TOTALLY DIFFERENT EFFECTS IN DIFFERENT LOCATIONS. (THIS PROBLEM IS EASED BY REX'S POLICY OF BUILT IN FUNCTIONS STARTING WITH "RX": THESE AT LEAST MAY HAVE SOME INTEGRITY).

4. THE CMS IMPLEMENTATION

4.1 INSTALLATION AND HELP: THE REX EXEC

THE BASIC REX PACK INCLUDES A (CMS) EXEC WHICH PERFORMS SEVERAL FUNCTIONS:

INSTALLATION:

"REX I" WILL INSTALL REX AS A NUCLEUS EXTENSION, AND WILL ALSO INSTALL EXEC 2 AND CPX IF NECESSARY (IE IF AVAILABLE AND NOT ALREADY ACTIVE). ONCE THIS HAS EXECUTED SUCCESSFULLY, YOU MAY RUN REX EXECS ON YOUR SYSTEM (AND CONTINUE TO RUN EXEC OR EXEC 2 PROGRAMS). THE FUNCTION PACKS (REXFNS ETC.) WILL BE LOADED AUTOMATICALLY BY REX AS AND IF REQUIRED.

YOU WILL PROBABLY WANT TO PUT THE LINE "EXEC REX I" INTO YOUR PROFILE EXEC.

GENERAL INFORMATION:

"REX ?" WILL TELL YOU ABOUT REX EXEC AND HOW TO USE THE TUTORIAL/HELP FACILITY.

TUTORIAL/ONLINE HELP FACILITY:

"REX" WILL TAKE YOU DIRECTLY TO THE INDEX OF THE TUTORIAL/HELP FACILITY - THEN SELECT TOPIC YOU WISH TO READ ABOUT BY NUMBER, KEYWORD, BY HITTING PF KEYS, OR BY USING THE LIGHTPEN.

A MAIN INDEX IS PRESENTED: THERE IS A SUB-INDEX FOR EACH OF IOX, FSX, REXFNS, REXFNS2, REXWORDS.

"REX XXXXX" (WHERE XXXXX IS A REX KEYWORD) WILL TAKE YOU DIRECTLY TO THE PART OF THE ONLINE DOCUMENTATION DESCRIBING THAT INSTRUCTION.

ERROR CODE INFORMATION:

YOU CAN GET EXTRA INFORMATION AND HINTS ON THE LIKELY CAUSES OF REX ERRORS BY TYPING "REX NNNNN" (WHERE NNNNN IS THE ERROR CODE, E.G. 20006).

4.2 SYSTEM INTERFACES

AS NOTED EARLIER, THE CURRENT REX IMPLEMENTATION USES THE YKTSVC PACKAGE, SINCE IT OFFERS THE BEST WAY OF 'GETTING INTO THE SYSTEM'. REX USES THE SAME INTERFACE CONVENTIONS AS EXEC 2 (NEW-FORM PLIST, ETC.) SO IT IS USABLE BY ANY PROGRAM, SUCH AS XEDIT, CURRENTLY ABLE TO INTERFACE WITH EXEC 2.

REX IS NORMALLY INSTALLED AS A NUCLEUS EXTENSION CALLED 'EXEC' AND THEREFORE INTERCEPTS ALL EXEC CALLS. IT THEN READS THE EXEC FILE (OR FILEBLOCK DEFINED DATA, SEE BELOW) UNTIL THE FIRST NON-BLANK CHARACTER IS MET. IF THE FIRST NON-BLANK CHARACTER IS '/' (IE THE START OF A REX COMMENT), THE FILE WILL BE ASSUMED TO BE WRITTEN IN THE REX LANGUAGE - OTHERWISE IT IS ASSUMED TO BE AN 'EXEC' OR 'EXEC 2' LANGUAGE FILE AND WILL BE PASSED ON AS APPROPRIATE:

IF A NUCX (NUCLEUS EXTENSION) CALLED EXEC2 EXISTS, AND THE CALL WAS AN EXEC 2 CONVENTIONAL CALL (OR THE FIRST WORD IN THE FILE WAS "&TRACE") THEN THE PLIST(S) WILL BE PASSED DIRECTLY TO IT FOR PROCESSING. OTHERWISE THE PLIST WILL BE PASSED DIRECTLY TO THE CMS EXEC PROCESSOR.

THESE SLIGHTLY MESSY RULES ALLOW REX PROGRAMS TO COEXIST AND BE USED SIMULTANEOUSLY WITH BOTH EXEC 2 AND EXEC PROGRAMS, AND IN ADDITION ALSO WORK IF EXEC 2 IS PART OF THE CMS NUCLEUS.

INTERNAL CALLS (FROM REX TO A USER COMMAND OR SUBCOMMAND) FOLLOW THE SAME CONVENTIONS AS EXEC 2 (NEW-FORM PLIST IS GENERATED, ETC.), EXCEPT THAT FUNCTION CALLS USE ONLY THE SIMPLE 'OLD FORMAT' PLIST TO REDUCE OVERHEAD. MICHEL HACK'S DOCUMENT "EXEC2SYS MEMO" IS THE BEST AND MOST AUTHORITATIVE SOURCE FOR FURTHER INFORMATION ON THE DETAILS OF THESE INTERFACES.

REX MAY BE CALLED WITH A "NEW-FORM PLIST" (IN ADDITION TO THE STANDARD CMS 8-BYTE TOKENISED PLIST) WHICH ALLOWS THE FOLLOWING POSSIBILITIES:

1) AN ARBITRARY PARAMETER STRING (NEITHER UPPER CASE, NOR TOKENISED) MAY BE PASSED TO REX.
2) A FILE OTHER THAN THAT DEFINED IN THE "OLD" PLIST MAY BE USED. (I.E. THE FILETYPE NEED NOT BE 'EXEC').
3) A DEFAULT TARGET FOR COMMANDS (OTHER THAN CMS). A FILETYPE OTHER THAN 'EXEC' OR ' ' WILL CAUSE COMMANDS TO GO TO THE ENVIRONMENT WITH THE NAME GIVEN.
4) A PROGRAM WHICH EXISTS IN STORAGE MAY BE EXECUTED (INSTEAD OF BEING READ FROM A FILE). THIS IN-STORAGE EXECUTION OPTION MAY BE USED FOR IMPROVED PERFORMANCE WHEN A REX PROGRAM IS BEING EXECUTED REPEATEDLY.

CALLING REX WITH A NEW-FORM PLIST:

BYTE 0 OF R1 = X'01' (SIGNIFIES NEW-FORM PLIST AVAILABLE)

R0 POINTS TO THE NEW-FORM PLIST:

```
**  THE NEW FORM PLIST, POINTS TO 1) THE ARGUMENT STRING, AND 2) AN
**  OPTIONAL FILE BLOCK:
NPLIST   DS    0F                  ** NEW-FORM PLIST
         DC    A(COMVERB)          -> CL5'EXEC '  (NAME OF INTERPRETER)
         DC    A(BEGARGS)          -> START OF ARGS STRING
         DC    A(ENDARGS)          -> CHARACTER AFTER END OF ARGS STRING
         DC    A(FBLOCK)           -> FILE BLOCK, OTHERWISE IS A(0)


** THE FILE BLOCK (ONLY REQUIRED IF REX IS TO EXECUTE A NON-EXEC FILE
**   OR IS TO EXECUTE FROM STORAGE)
FBLOCK   DS    0F                  ** FILE BLOCK
         DC    CL8'FILENAME'       LOGICAL NAME OF PROGRAM
         DC    CL8'FILETYPE'       DEFAULT DESTINATION FOR COMMANDS
                                     (BLANKS, "EXEC" OR " " BOTH CAUSE
                                     COMMANDS TO BE PASSED TO CMS)
         DC    CL2'FILEMODE'       SHOULD NORMALLY BE '* ' OR '  '
         DC    H'EXTLEN'           LENGTH OF EXTENSION BLOCK IN FULL-
                                     WORDS: H'2' IF IN-STORE EXECUTION
                                     IS DESIRED, H'0' OTHERWISE.
*        EXTENSION BLOCK STARTS HERE (ONLY REQUIRED FOR IN-STORAGE PGM)
         DC    AL4(PGMFILE)        -> START OF PROGRAM DESCRIPTOR LIST
         DC    AL4(PGMEND-PGMFILE) LENGTH OF PROGRAM DESCRIPTOR LIST


** DESCRIPTOR LIST FOR IN-STORAGE PROGRAM
PGMFILE  DS    0F                  ** IN STORAGE REX/EXEC 2 PROGRAM
         DC    A(LINE1),F'LEN1'    ADDRESS, LENGTH OF LINE 1
         DC    A(LINE2),F'LEN2'    ADDRESS, LENGTH OF LINE 2
            ...
         DC    A(LINEN),F'LENN'    ADDRESS, LENGTH OF LINE N
PGMEND   EQU   *
```

NOTES:

THE IN-STORAGE PROGRAM LINES NEED NOT BE CONTIGUOUS, SINCE EACH IS SEPARATELY DEFINED IN THE DESCRIPTOR LIST.

FOR IN-STORE EXECUTION, FILENAME AND FILETYPE ARE STILL REQUIRED IN THE FILE BLOCK, SINCE THESE DETERMINE THE LOGICAL PROGRAM NAME AND THE DEFAULT COMMAND ENVIRONMENT.

4.3 STANDARD FUNCTION PACKS

REX INCLUDES THREE BASIC PACKS OF FUNCTIONS: REXFNS, REXFNS2, AND REXWORDS. THE PACKS WILL BE LOADED AUTOMATICALLY BY REX IF AND WHEN NEEDED, OR THEY MAY BE EXPLICITLY LOADED BY ISSUING THEIR NAME AS A COMMAND. ALL THREE "TELL" ABOUT THEMSELVES WHEN INVOKED WITH THE ARGUMENT "?". (NOTE: REXFNS2 STACKS A COMMAND WHICH INVOKES IOS3270 FOR THIS FUNCTION).

4.3.1 REXFNS

THE BASIC SET OF EXEC-LIKE FUNCTIONS ARE INCLUDED IN THE 'REXFNS' PACKAGE.

DATATYPE(STRING)	RETURNS 'NUM' IF THE STRING IS A VALID NUMBER OTHERWISE RETURNS 'CHAR'.
SYMBOL('NAME')	RETURNS 'VAR' IF THE SYMBOL NAMED HAS BEEN ASSIGNED A VALUE, OTHERWISE RETURNS 'LIT'.
TRANS(STRING,TABO,TABI)	PL/I-LIKE TRANSLATE FUNCTION. IF NEITHER TRANSLATE TABLE IS GIVEN, THE STRING IS TRANSLATED TO UPPER CASE. TABI IS INPUT TRANSLATE TABLE (DEFAULT IS FULL EBCDIC SET), AND TABO IS OUTPUT TABLE (WHICH IS PADDED WITH BLANKS OR TRUNCATED AS NECESSARY).
LENGTH(STRING)	RETURNS THE LENGTH OF THE STRING
WCOUNT(STRING)	RETURNS THE NUMBER OF WORDS IN THE STRING
INDEX(HAYSTACK,NEEDLE)	RETURNS THE POSITION OF THE NEEDLE IN THE HAYSTACK (SAME FORMAT AS PL/I).
POS(NEEDLE,HAYSTACK)	ALSO RETURNS THE POSITION OF THE NEEDLE IN THE HAYSTACK (NOT AS IN PL/I).
LASTPOS(NEEDLE,HAYSTACK)	RETURNS THE POSITION OF THE LAST OCCURRENCE OF THE NEEDLE IN THE HAYSTACK.
REPEAT(STRING,N)	RETURNS N+1 CONCATENATED COPIES OF STRING
SUBSTR(STRING,N,K)	RETURNS THE SUBSTRING OF STRING WHICH BEGINS AT THE NTH CHARACTER, AND IS OF LENGTH K, PADDED WITH BLANKS IF NECESSARY.
DELSTR(STRING,N,K)	DELETES THE SUBSTRING OF STRING WHICH BEGINS AT THE NTH CHARACTER, AND IS OF LENGTH K.
TRUNC(STRING,N)	RETURNS INTEGER PART OF NUMBER, AND UP TO N DIGITS AFTER THE "." (DEFAULT N=0). (NO PAD)
READFLAG()	RETURNS 'CONSOLE' OR 'STACK' DEPENDING ON FROM WHERE THE NEXT "PULL" WILL READ.
NEST()	RETURNS CURRENT DEPTH OF NESTING OF EXECS (INCLUDING CMS AND EXEC 2 EXECS)
USERID()	RETURNS THE VIRTUAL MACHINE USERID.
DATE()	RETURNS THE DATE E.G. '8 OCT 79'
TIME()	RETURNS THE LOCAL TIME E.G. '04:41:37'

4.3.2 REXFNS2

REXFNS2 INCLUDES A NUMBER OF HIGH LEVEL CHARACTER MANIPULATION FUNCTIONS, TOGETHER WITH A USEFUL SET OF CONVERSION ROUTINES ETC.

NOTE: FOR FUNCTIONS WHICH PROVIDE FOR A "PAD" CHARACTER, THE PAD CHARACTER IS OPTIONAL: IF SPECIFIED, THE SHORTER STRING IS EXTENDED ON THE RIGHT WITH THE PAD CHARATER, OTHERWISE, THE OPERATION IS PERFORMED ONLY ON THE PORTIONS OF THE STRINGS WHICH CORRESPOND IN LENGTH.

AND(STRING1,STRING2,PAD)	RETURNS THE LONGER OF THE TWO STRINGS, WITH WHICH THE SHORTER HAS BEEN LOGICALLY ANDED.
CLCL(STRING1,STRING2,PAD)	LOGICALLY COMPARES THE TWO STRINGS AND RETURNS THE POSITION OF THE FIRST CHARACTERS WHICH MISCOMPARE (ZERO IF THE STRINGS ARE EQUAL, NEGATIVE IF STRING1 IS LESS THAN STRING2.
CLXL(STRING1,STRING2,PAD)	LIKE CLCL EXCEPT THAT THE COMPARISON IS ARITHMETIC, AND THE STRINGS ARE HEXADECIMAL.
COUNTBUF()	RETURNS NUMBER OF RECORDS IN CONSOLE STACK.
D2X(NUMBER,K)	RETURNS A CHARACTER STRING OF LENGTH "K" ("K" IS BETWEEN 1 AND 8) WHICH IS THE HEXADECIMAL REPRESENTATION OF THE DECIMAL INTEGER "NUMBER". IF "K" IS OMITTED, LEADING ZEROS ARE SUPRESSED.
E2X(STRING)	SAME AS THE REX NOTATION 'STRING'X, EXCEPT THAT "STRING" MAY BE A VARIABLE. IE 'PACK'.
FETCH(ADDRESS,K)	RETURNS THE CONTENTS OF "K" BYTES OF THE USER'S VIRTUAL MEMORY STARTING AT "ADDRESS". BOTH "ADDRESS" AND "K" ARE PACKED HEXA-DECIMAL VALUES. EG '020000'X.
LOCATE(NEEDLE,HAYSTACK,N,'¬')	RETURNS THE POSITION OF THE N'TH OCCURRENCE OF "NEEDLE" IN "HAYSTACK" (OR, IF '¬' IS SPECIFIED, THE N'TH OCCURRENCE OF ANY STRING IN "HAYSTACK" WHICH IS EQUAL IN LENGTH TO "NEEDLE" BUT WHICH IS NOT EQUAL TO "NEEDLE"). IF "N" IS NEGATIVE, THE SEARCH IS RIGHT TO LEFT. (NOTE: THIS FUNCTION IS DISK RESIDENT, NOT IN REXFNS2).
OR(STRING1,STRING2,PAD)	RETURNS THE LONGER OF THE TWO STRINGS, WITH WHICH THE SHORTER HAS BEEN LOGICALLY ORED.
REVERSE(STRING)	RETURNS "STRING", SWAPPED END FOR END.
SUBSET()	RETURNS 1 IF IN SUBSET, 0 OTHERWISE.
TM(STRING,MASK,PAD)	THE BITS OF "STRING" ARE TESTED BYTE FOR BYTE UNDER THE "1" BITS OF "MASK" (ONLY "MASK" MAY BE EXTENDED WITH THE PAD CHARACTER). ZERO IS RETURNED IF ALL BITS TESTED WERE ZERO; -1 IS RETURNED IF ALL BITS TESTED WERE ONE; OTHERWISE, THE POSITION OF THE FIRST CHARACTER IN "STRING" WHICH CAUSED A "MIXED ONES AND ZEROS" CONDITION IS RETURNED.
TRT(STRING,REFERENCE,¬)	RETURNS THE POSITION OF THE 1ST CHARACTER IN "STRING" WHICH IS NOT ALSO IN "REFERENCE", OR, IF '¬' IS SPECIFIED, THE POSITION OF THE FIRST CHARACTER IN "STRING" WHICH IS IN "REFERENCE".

```
TYPEFLAG(HT OR RT)          RETURNS HT OR RT. WILL ALSO HALT TYPING OR
                              RESUME TYPING IF HT OR RT IS SPECIFIED.
VERIFY(STRING,REFERENCE,¬)  SYNONYM FOR TRT ABOVE.
XOR(STRING1,STRING2,PAD)    RETURNS THE LONGER OF THE TWO STRINGS, WITH
                              WHICH THE SHORTER HAS BEEN LOGICALLY XORED.
XRANGE(M,N)                 RETURNS A STRING OF ALL ONE BYTE CODES
                              BETWEEN THE VALUES "M" AND "N".
X2D(HEXSTRING)              CONVERTS "HEXSTRING" TO DECIMAL.
X2E(HEXSTRING)              RETURNS THE EBCDIC CHARACTER REPRESENTATION
                              OF "HEXSTRING". IE. UNPACKS.
```

4.3.3 REXWORDS

SOME USEFUL 'WORD PROCESSING' FUNCTIONS ARE IN THE 'REXWORDS' PACK. ALL THESE FUNCTIONS NORMALISE THE STRING FIRST (REMOVE LEADING AND TRAILING BLANKS, AND REDUCE INTER-WORD GAPS TO ONE SPACE).

```
LEFT(STRING,K)        RETURNS A STRING OF LENGTH K WITH 'STRING' LEFT
                        JUSTIFIED AND PADDED WITH BLANKS IF NEEDED,
RIGHT(STRING,K)       RETURNS A STRING OF LENGTH K WITH 'STRING' RIGHT
                        JUSTIFIED AND PADDED WITH BLANKS IF NEEDED,
CENT(STRING,K)        RETURNS A STRING OF LENGTH K WITH 'STRING' CENTRED
                        CENTERED IN IT.  E.G:  CENT(ABC,7) => '  ABC  '
JUSTIFY(STRING,K)     FORMATS THE WORDS IN THE STRING, BY ADDING BLANKS
                        BETWEEN WORDS, TO JUSTIFY TO BOTH MARGINS.
SPACE(STRING,N)       FORMATS THE WORDS IN THE STRING WITH N SPACES
                        BETWEEN WORDS.  N MAY BE 0, TO REMOVE ALL BLANKS.
WORD(STRING,N)        RETURNS THE NTH WORD IN THE STRING. E.G:
                        WORD('NOW IS THE TIME',3) => 'THE'
FIND(STRING,PHRASE)   FIND THE WORD NUMBER OF THE FIRST WORD IN THE
                        PHRASE.  RETURNS 0 IF THE PHRASE IS NOT FOUND.
                        E.G:  FIND(NOW IS THE TIME,IS THE) => '2'
```

4.4 INTERFACE TO FUNCTIONS

IN CMS, REX FUNCTIONS ARE CALLED VIA SVC 202, USING A SPECIAL SEARCH ORDER (SEE THE DIAGRAM AT THE END OF THIS SECTION): UNLESS THE FUNCTION NAME IS 'EXEC', REX WILL PREFIX 'RX' TO THE FUNCTION NAME GIVEN (TRUNCATING ON THE RIGHT IF NECESSARY) AND WILL THEN ATTEMPT TO EXECUTE THE RESULTING FUNCTION USING SVC 202. IF THE FUNCTION IS NOT FOUND, THEN THE FUNCTION PACKS WILL BE INTERROGATED AND LOADED IF NECESSARY (THEY RETURN RC=0 IF THEY CONTAINED THE REQUESTED FUNCTION, OR RC=1 OTHERWISE). IF STILL NOT FOUND, THEN THE NAME IS RESTORED TO ITS ORIGINAL FORM AND RE-EXECUTED WITH SVC 202 (IF STILL NOT FOUND, AN ERROR IS RAISED). THIS MECHANISM ALLOWS REX FUNCTIONS TO BE WRITTEN WITH LITTLE CHANCE OF NAME CONFLICT WITH EXISTING MODULES.

WHEN THE FUNCTION RECEIVES CONTROL, REGISTER 1 POINTS TO A NORMAL CMS PLIST, AND THE TOP BYTE OF R1 IS X'00'. REGISTER 0 POINTS TO A LIST OF ARGUMENT DESCRIPTORS, BEING A SERIES OF FULLWORD PAIRS. THE FIRST VALUE IN EACH PAIR IS THE ADDRESS OF THE ARGUMENT CHARACTER STRING, AND THE SECOND VALUE IS IT'S LENGTH (WHICH MAY BE 0). THE FINAL VALUE PAIR IS FOLLOWED BY TWO FULLWORDS CONTAINING F'-1' (IE. X'FFFFFFFF'). CURRENTLY THERE IS A MAXIMUM OF SEVEN ARGUMENTS ENFORCED.

DURING CALCULATION OF THE RESULT, THE FUNCTION MAY USE THE ARGUMENTS (WHICH RESIDE IN USER STORAGE OWNED BY REX) AS WORK AREAS, WITHOUT FEAR OF CORRUPTING INTERNAL REX VALUES.

THE RESULT MUST BE RETURNED TO REX IN A BLOCK OF STORAGE ALLOCATED BY DMSFREE AND WHICH HAS THE FOLLOWING STORAGE ASSIGNMENTS AND VALUES:

```
*-- DSECT FOR THE RETURNED DATA BLOCK ------------------------*
EVALBLOK DSECT
EVBPAD1  DS    F                  RESERVED
EVSIZE   DS    F                  TOTAL BLOCK SIZE IN DW'S
EVLEN    DS    F                  LENGTH OF DATA (IN BYTES)
EVBPAD2  DS    F                  RESERVED
EVDATA   DS    C...               THE RETURNED CHARACTER STRING
```

THE ADDRESS OF THIS BLOCK SHOULD BE STORED IN THE FIRST FULLWORD OF THE ARGUMENT LIST (I.E. THE LOCATION POINTED TO BY REGISTER 0 ON ENTRY TO THE FUNCTION).

THIS BLOCK WILL ONLY BE ACCEPTED (AND LATER FREED) BY REX IF THE FUNCTION RETURNS A ZERO 'RETURN CODE' IN REGISTER 15.

THIS INTERFACE HAS THREE MAJOR ADVANTAGES:

THERE IS NO RESTRICTION ON THE CONTENT OF THE DATA RETURNED.

THERE IS NO RESTRICTION (OTHER THAN YOUR VM SIZE) ON THE LENGTH OF DATA RETURNED.

THE RETURNED BLOCK IS IMMEDIATELY USABLE BY REX, WITHOUT NEED TO COPY THE DATA.

USING THE STACK WOULD REQUIRE TWO INVOCATIONS OF THE STACK HANDLING ROUTINES FOR EACH ARGUMENT AND RESULT. THIS OVERHEAD IS AVOIDED.

CALLING EXECS AS FUNCTIONS WORKS THE SAME WAY, WITH THE FOLLOWING POINTS BEING RELEVANT:

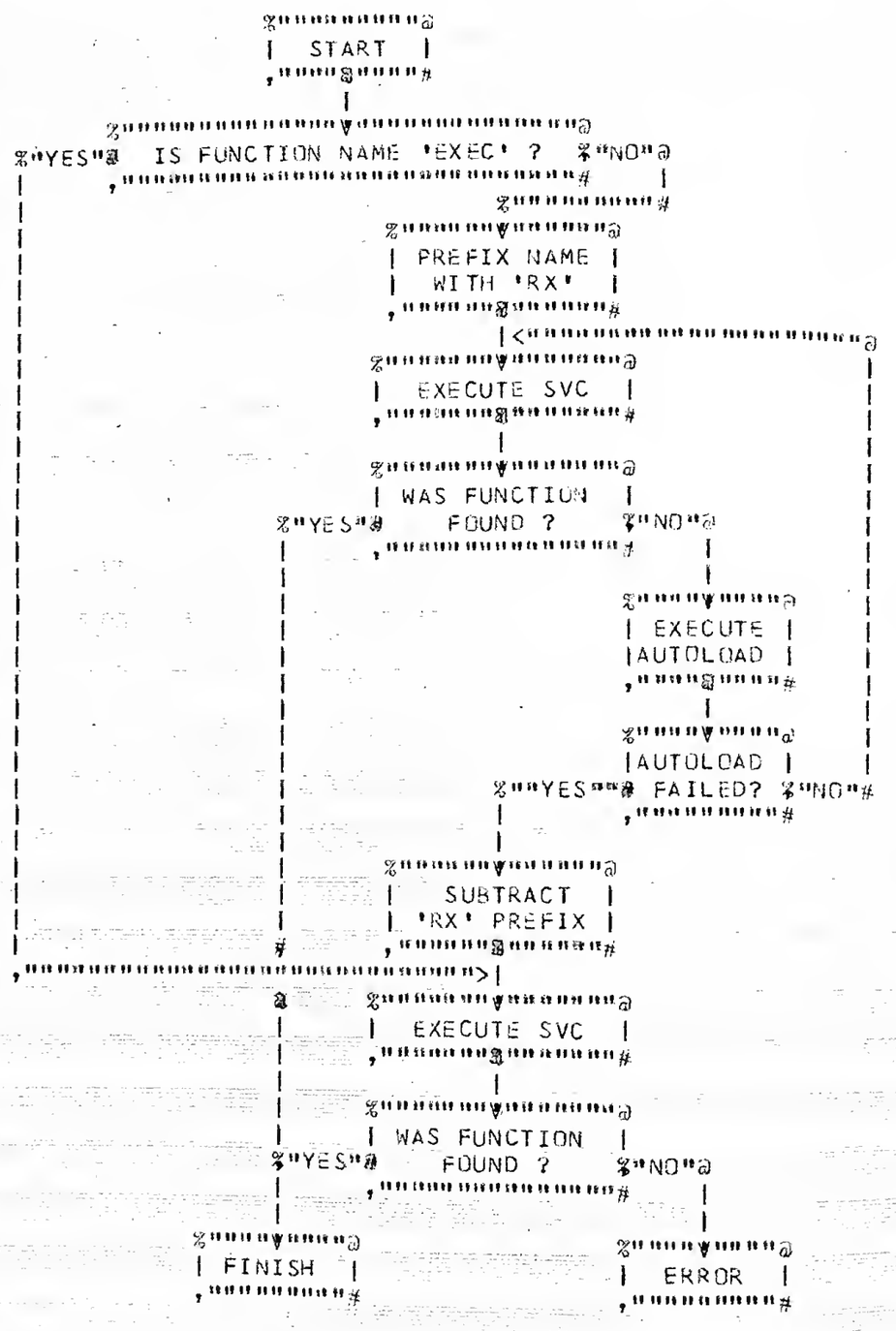
1) ONLY ONE ARGUMENT STRING IS USED BY REX (OTHERS ARE IGNORED)
2) THE ARGUMENT STRING IS RESTRICTED TO 240 BYTES.
3) THE RETURN INSTRUCTION WILL PASS BACK A REX EVALBLOK DIRECTLY, INSTEAD OF USING THE STACK. THERE IS THEREFORE NO RESTRICTION ON THE LENGTH OR CONTENT OF THE DATA RETURNED.
4) THE SPECIAL PROCESSING INVOLVED IN THIS IS TRANSPARENT TO THE USER.
5) YOU MAY INVOKE AN EXEC AS A FUNCTION USING THE SYNTAX:
   'EXEC FRED'(ARGSTRING)
   SINCE THIS IS RATHER CUMBERSOME, YOU MAY PREFER TO ASSIGN THE LITERAL TO A VARIABLE FIRST FOR USE AS REQUIRED, THUS:
   FR='EXEC FRED'; /* NOW "FR(ARGSTRING)" WILL WORK */

IMPLEMENTATION NOTE: THE THREE MAIN FUNCTION PACKS ALSO RESPOND TO A CALL OF THE FORM:

```
REXNAME  LOAD  RXFNAME
```

IF RXFNAME IS CONTAINED WITHIN THE PACK REXNAME, THEN IT WILL NUCXLOAD ITSELF AND RETURN RC=0, OTHERWISE RC=1 IS RETURNED. THIS ALLOWS THE FUNCTION PACKS TO BE AUTOMATICALLY LOADED BY REX WHEN NECESSARY.

REX FUNCTION RESOLUTION AND EXECUTION:



4.5 DIRECT INTERFACE TO REX VARIABLES

(NOTE: THIS SECTION DESCRIBES THE INTERFACE FOR ALL REX VERSIONS SINCE 1.12. EARLIER VERSIONS HAD A DIFFERENT AND MORE COMPLICATED CALLING MECHANISM, AND A MORE RESTRICTED 16-BYTE PLIST. COPIES OF IOX, FSX, ETC. PRIOR TO MARCH 1980 ARE INCOMPATIBLE WITH CURRENT RELEASES.)

REX (UNDER CMS) PROVIDES AN INTERFACE WHEREBY CALLED COMMANDS MAY EASILY ACCESS AND MANIPULATE THE CURRENT GENERATION OF REX VARIABLES. VARIABLES MAY BE INSPECTED, SET, OR DROPPED; AND IF REQUIRED ALL THE ACTIVE VARIABLES MAY BE INSPECTED IN TURN. ALL MANIPULATION OF INTERNAL REX WORK AREAS IS CARRIED OUT BY REX'S ROUTINES: USER PROGRAMS DO NOT THEREFORE NEED TO KNOW ANYTHING OF THE STRUCTURE OF THE VARIABLES ACCESS METHOD (WHICH INCLUDES HASH TABLES, ETC. ETC.).

THE INTERFACE WORKS AS FOLLOWS:

WHEN REX STARTS TO INTERPRET A NEW EXEC OR EDITOR MACRO IT FIRST SETS UP A SUBCOMMAND ENTRY POINT CALLED "EXECCOMM". WHEN A PROGRAM (COMMAND OR SUBCOMMAND) IS INVOKED BY REX, IT MAY IN TURN USE THE CURRENT EXECCOMM ENTRY POINT TO SET, READ, OR DROP REX VARIABLES USING REX'S INTERNAL ROUTINES.

THE INTERNAL ROUTINE REXVAR CARRIES OUT ALL CHANGES TO POINTERS, ALLOCATION OF STORAGE, SUBSTITUTION OF VARIABLES IN THE NAME, ETC. HENCE ISOLATING USER PROGRAMS FROM THE INTERNAL MECHANISMS OF REX.

TO ACCESS VARIABLES, A PLIST MUST BE SET UP (SEE BELOW), AND THEN AN SVC 202 SHOULD BE ISSUED (WITH R1 POINTING TO THE PLIST, AND THE TOP (FLAG) BYTE OF R1 SET TO X'02').

R0 MUST BE SET TO 0. FUTURE EXTENSION MAY POSSIBLY USE R0 TO POINT TO A NORMAL NEW-FORM PLIST WHICH WILL CONTAIN A STRING TO BE INTERPRETED.

ON RETURN FROM THE SVC, R15 WILL CONTAIN THE RETURN CODE FROM THE ENTIRE PLIST. THE POSSIBLE RETURN CODES ARE:

R15 =	0	ENTIRE PLIST PROCESSED SUCCESSFULLY
	-1	INVALID FUNCTION REQUEST
	-3	(FROM SUBCOM) NO EXECCOMM ENTRY POINT FOUND: IE NOT CALLED FROM INSIDE A REX EXEC.
	5	INSUFFICIENT FREE STORAGE AVAILABLE TO EXECUTE
	30	A NAME (FOLLOWING ANY SUBSTITUTION) WAS TOO LONG (>150)
	31	NAME MAY BE NUMERIC (FIRST CHARACTER '0'-'9' OR '.') NOTE THAT A VARIETY OF OTHER NAMES, ILLEGAL FOR USE WITHIN REX, MAY BE USED BY THIS INTERFACE: THIS ALLOWS PROGRAMS TO USE 'SAFE' NAMES.

NOTES:

REXVAR RUNS WITH INTERRUPTS ENABLED FOR MOST OF THE TIME.

THE INTERFACE MAY BE DISABLED BY SETTING THE FIRST BYTE OF THE UWORD IN THE SCBLOCK TO X'00'. REX WILL THEN REPLY 'INVALID FUNCTION REQUEST' (R15=-1) IF ANY COMMAND TRIES TO ACCESS VARIABLES. RESTORING THE BYTE

TO X'FF' WILL RE-ENABLE THE INTERFACE.

THE PLIST:

THE PLIST MUST CONSIST OF THE EIGHT BYTE STRING 'EXECCOMM' FOLLOWED BY ANY NUMBER OF CONSECUTIVE 20-BYTE BLOCKS (SO ALLOWING MULTIPLE OPERATIONS) THUS:

```
 REXENTPL DSECT
 REXENTNM DS   CL8     MUST BE SET TO 'EXECCOMM', THE SUBCOMMAND NAME.
 * THE NEXT 20 BYTES MAY BE REPEATED AS OFTEN AS REQUIRED
 REXENTFN DS   CL1     PLIST FUNCTION CODE:
 *                       C'D'  = DROP VARIABLE
 *                       C'Q'  = QUERY (READ) VARIABLE
 *                       C'N'  = QUERY NEXT LOGICAL VARIABLE (SEE BELOW)
 *                       C'S'  = SET VARIABLE
 *                       X'FF' = END OF PLIST
 *
 REXENTFG DS   CL1     FUNCTION RETURN CODE (SET BY REX)
 *                       X'00' = OPERATION COMPLETED OK
 *                       X'01' = VARIABLE DID NOT EXIST BEFORE OPERATION
 *                         FOR:  DROP:  NO OTHER ACTION TAKEN
 *                               QUERY: NO OTHER ACTION TAKEN
 *                               SET:   VARIABLE CREATED AND SET
 *                       X'02' = (AFTER "N" FUNCTION ONLY) LAST VARIABLE
 *                               HAS BEEN QUERIED.
 *
          DS   CL2     RESERVED FOR FUTURE USE, SHOULD BE SET TO X'0000'
 REXENTNA DS   AL4     ADDRESS OF VARIABLE NAME BUFFER
 REXENTNL DS   F       LENGTH OF VARIABLE NAME (CURRENT MAX 150)
 REXENTDA DS   AL4     ADDRESS OF DATA BUFFER
 REXENTDL DS   F       LENGTH OF DATA BUFFER
 * REXENTDA & REXENTDL:  SHOULD BE SET BY THE USER IF FUNCTION='S'
 *          ...          ARE SET BY REX  IF FUNCTION='Q' OR 'N'
 *          ...            AND REXENTFG=X'00'
 *          ...          ARE NOT ALTERED IF REXENTFG IS NOT X'00'
 *          ...          ARE NOT USED IF FUNCTION='D'
```

*--

A TYPICAL CALLING SEQUENCE (FULLY RELOCATABLE (NUCXLOADABLE) AND READ-ONLY CODE) MIGHT BE:

```
          XR    R0,R0                CLEAR R0
          LA    R1,UPLIST            -> USER PLIST, AS ABOVE
          ICM   R1,B'1000',=X'02'    INSERT "SUBCOMMAND CALL" FLAG
          CNOP  2,4                  ALIGNMENT
          LA    R14,*+10             PREPARE ERROR EXIT ADDRESS
          SVC   202                  ISSUE SVC
          DC    F'1100'              ADDRESS OF "BR 14" IN NUCON
          LTR   R15,R15              TEST RETCODE
          BNZ   VARERROR             QUIT IF WE HAD A CATASTROPHIC ERROR
 *        EXECUTION WAS OK
```

THE FUNCTION 'QUERY NEXT LOGICAL VARIABLE' MAY BE USED TO SEARCH THOUGH ALL THE VARIABLES KNOWN BY REX (AT THE CURRENT LEVEL).

REX MAINTAINS POINTERS TO ITS LIST OF VARIABLES: THESE ARE RESET WHENEVER 1) A HOST COMMAND IS ISSUED, OR 2) ANY FUNCTION OTHER THAN 'N' IS EXECUTED VIA THIS DIRECT VARIABLES INTERFACE.

WHENEVER AN 'N' (NEXT) FUNCTION IS EXECUTED, REX WILL RETURN THE LENGTHS AND ADDRESSES OF THE NAME AND DATA PERTAINING TO THE NEXT LOGICAL VARIABLE KNOWN TO IT. BY REPEATEDLY EXECUTING 'N' (UNTIL THE RETURN FLAG IS SET TO X'02' IN THE PLIST) A USER PROGRAM MAY LOCATE ALL THE CURRENTLY ACTIVE REX VARIABLES. THE ADDRESSES POINT TO THE 'REAL' DATA, SO THE USER MAY ALTER THE DATA IF DESIRED, THOUGH THE LENGTH CANNOT BE CHANGED.

THE RETURN CODE FROM FUNCTION 'N' WILL BE 0 OR NEGATIVE. IF 0, THEN IF REXENTFG IS SET TO X'00' THEN THIS INDICATES THAT A VARIABLE HAS BEEN FOUND AND THE POINTERS AND LENGTHS HAVE BEEN PUT IN THE PLIST. X'02' IN REXENTFG INDICATES THAT THE LAST VARIABLE HAS BEEN READ: THE INTERNAL POINTERS HAVE BEEN RESET, AND NO VALID DATA HAS BEEN STORED IN THE PLIST.

IN THIS MANNER A PROGRAM (SUCH AS THE 'REXDUMP' DEBUG AID) MAY INSPECT ALL THE VARIABLES CURRENTLY KNOWN.

4.6 USING SERVICE PROGRAMS WITH REX (IOX, FSX, ETC.)

ALL COMMANDS THAT MAY BE CALLED FROM EXEC OR EXEC 2 MAY BE USED WITH REX, EXCEPT THAT THOSE WHICH ATTEMPT TO SET 'OLD EXEC' VARIABLES MAY NOT WORK. SOME MODULES WHICH ARE ESPECIALLY USEFUL ARE:

IOX: IS A SERVICE PROGRAM WHICH WAS WRITTEN ESPECIALLY FOR USE WITH REX AND IS RECOMMENDED. IT MAY BE USED TO READ, WRITE, UPDATE, AND SEARCH FILES; PRINT OR PUNCH RECORDS; SET GLOBAL VARIABLES; ETC.

FSX: IS A REX SERVICE PROGRAM DESIGNED TO GIVE USERS COMPLETE CONTROL OF FULL SCREEN DISPLAYS (E.G. FOR MODELLING FUTURE APPLICATIONS).

STACKIO: HAS VERY MANY USEFUL FEATURES AND IS FULLY COMPATIBLE WITH REX, HOWEVER ITS I/O IS LIMITED TO THE WIDTH OF THE STACK.

EMSG: HAS THE SAME EFFECT AS &EMSG IN CMS EXECS.

IOS3270: RECENT VERSIONS WILL (I UNDERSTAND) DETECT WHEN THEY ARE CALLED FROM EXEC 2 OR REX. OLDER MODULES MAY FAIL TO RUN SATISFACTORILY. THE EXEC VARIABLES &IOSK ETC. WILL NOT OF COURSE BE SET WHEN IOS3270 IS CALLED FROM EXEC 2 OR REX, NOR MAY VARIABLES BE INCLUDED IN THE DATA STREAM. A NEW VERSION, COMPATIBLE WITH THE REX VARIABLES INTERFACE, IS EXPECTED VERY SOON.

MODULES: CHECKS WHETHER LISTED MODULES EXIST ON ANY DISK: A MESSAGE IS TYPED FOR ANY THAT COULD NOT BE FOUND.

EXSERV: ALL FUNCTIONS OTHER THAN THOSE WHICH ATTEMPT TO SET EXEC VARIABLES SHOULD BE USEABLE.

GLOBALV: ALL FUNCTIONS OTHER THAN THOSE WHICH ATTEMPT TO SET EXEC VARIABLES SHOULD BE USEABLE.

PROMPT: A VERSION IS AVAILABLE WHICH WILL USE THE NEW FORMAT PLIST PROVIDED BY REX/EXEC 2. IT PROMPTS THE USER WITH DATA ON LINE 23.

REXDUMP: A DEBUG AID WHICH "DUMPS" UP TO 60 CHARACTERS OF EACH VARIABLE, AND THE LENGTH OF THE VARAIABLE, TO THE SCREEN.

DRAINSTK: PURGES ALL BUFFERS FROM THE INPUT CONSOLE STACK WITHOUT AFFECTING THE OUTPUT STACK.

HT, RT: HALT/RESUME TYPING. SAME AS STACKING HT, RT IN CMS EXEC.

OSRESET: A MODULE WHICH RESETS OS SIMULATED STORAGE. SHOULD BE INVOKED BETWEEN PL/I MODULE CALLS, FOR EXAMPLE, WHICH CAN OTHERWISE FAIL WITH A "VIRTUAL STORAGE SIZE EXCEEDED" MESSAGE. (ALSO USEFUL WITHIN EXEC 2 EXECS FOR THE SAME PURPOSE).

4.1 REX PROGRAM STRUCTURE

THE FOLLOWING INFORMATION MAY BE OF INTEREST TO SOME READERS:

REX IS IMPLEMENTED AS EIGHT CSECTS WHICH TOGETHER FORM A READ-ONLY MODULE THAT IS FULLY RELOCATABLE AND RECURSIVE. ALL 'SYSTEM DEPENDENT' CODE IS CONTAINED IN TRIVIAL MACROS, SO IN THEORY REX MAY BE MODIFIED FOR RUNNING UNDER A DIFFERENT OPERATING SYSTEM BY JUST REWRITING REX ASSEMBLE AND REX MACRO, THEN RE-ASSEMBLING THE OTHER CSECTS.

BRIEFLY, THE APPROXIMATE SIZE (IN SOURCE LINES + COMMENTS) AND FUNCTION OF EACH CSECT IS:

REX	- 1030	- READS THE EXEC FILE AND CALLS REXINT. ALSO HANDLES THE DIRECT INTERFACE TO VARIABLES.
REXCONV	- 380	- CONVERSION (NUMERIC <--> BINARY) ROUTINES.
REXEVAL	- 1680	- THE EXPRESSION EVALUATOR.
REXEXEC	- 2220	- EXECUTES THE INDIVIDUAL INSTRUCTION CLAUSES.
REXINT	- 1680	- PARSES THE INPUT DATA, CONTROLS MOST EXECUTION DECISIONS, AND PASSES CLAUSES FOR EXECUTION TO REXEXEC.
REXSAY	- 160	- SAY (DISPLAY ON TERMINAL) ANY LENGTH DATA.
REXTRACE	- 320	- FORMAT AND DISPLAY TRACE INFORMATION
REXVAR	- 630	- ACCESS REX VARIABLES, USING HASH TABLE ETC.

REX MACRO IS ABOUT 1110 LINES (60% OF WHICH ARE COMMENTS ETC.).

THE REX PACKAGE INCLUDES OTHER FILES AND UTILITIES, OF COURSE, AND THE APPROXIMATE SIZE OF THE MORE IMPORTANT OF THESE IS (AGAIN, IN LINES):

REXFNS	- 1260	- STANDARD BUILT-IN FUNCTIONS
REXWORDS	- 790	- WORD PROCESSING BUILT-IN FUNCTIONS
REX EXEC	- 150	- ON-LINE INSTALLATION AND DOCUMENTATION CONTROL
IOSLIB	- 2640	- ON-LINE DOCUMENTATION
SCRIPT	- 2800	- (THIS DOCUMENT)

4.8 WRITING BILINGUAL EXECS

IN SOME CIRCUMSTANCES IT MAY BE DESIRABLE TO WRITE EXECS THAT WILL RUN WHETHER OR NOT REX IS INSTALLED.

TO PERMIT THIS, REX ALLOWS ITS PROGRAMS TO START WITH '*/*' RATHER THAN '/*': BOTH THESE ALTERNATIVES ARE TAKEN TO BE THE START OF A COMMENT IF PARSED BY REX. IF THE FILE IS EXECUTED BY EXEC BECAUSE REX IS NOT INSTALLED, THEN THIS FIRST LINE WILL BE INTERPRETED AS A COMMENT BY IT TOO: SUBSEQUENT LINES MAY THEN CONTAIN 'OLD' EXEC LANGUAGE STATEMENTS.

E.G:

```
*/* THIS IS A TRIVIAL BILINGUAL EXEC
    &GOTO -OLD   */
SAY 'THIS IS EXECUTED WHEN REX IS INSTALLED'
EXIT

-OLD
&TYPE THIS IS EXECUTED BY EXEC WHEN REX IS NOT INSTALLED
```

THE TECHNIQUE MAY BE USED TO ALLOW AN EXEC TO BE WRITTEN IN REX WHICH HAS STATEMENTS AT THE START TO INSTALL REX AND RE-INVOKE THE EXEC IF REX IS NOT ALREADY ACTIVE. THE FOLLOWING SEQUENCE MAY BE USED AFTER THE LABEL -OLD ABOVE TO ACHIEVE THIS:

```
&CONTROL OFF
EXEC REX I
&IF &RETCODE EQ C EXEC &0 &1 &2 &3 &4 &5 &6 &7 ...
&EXIT &RETCODE
```

NOTE THAT EXEC 2/REX BILINGUAL EXECS ARE NOT POSSIBLE SINCE THE MEANS BY WHICH THEY ARE RECOGNISED ARE MUTUALLY EXCLUSIVE.

5. EXAMPLE EXECS FOR CMS USING REX

ADDR EXEC

```
/* DISPLAYS FULL ADDRESS AND NAME FOR NICKNAMES SPECIFIED */

ARGS ENAME REST
IF REST='' | REST=? THEN SIGNAL TELL
REST=TRANS(REST) /* ENSURE UPPERCASE */
DO I=1 TO WCOUNT(REST) /* FOR EACH WORD IN REST .. */
  PARSE VAR REST NICKNAME REST  /* GET 1ST TOKEN INTO NICKNAME */
  STATE NICKNAME DISTRIB '*'
  IF RC=0 THEN DO
    SAY NICKNAME 'IS A DISTRIBUTION LIST'
    ITERATE I
    END
  /* NOT A LIST */
  SCANRMSG NICKNAME
  IF RC=0 THEN /* SOME DATA WAS STACKED */ DO
    PULL NN NODE UID VIA N1 N2 N3 N4 N5
    IF UID='%' THEN
      SAY NICKNAME 'IS THE NICKNAME FOR THE LOCAL USER' VIA
     ELSE
      SAY NICKNAME 'IS THE NICKNAME FOR' N1 N2 '('UID AT NODE')'
    ITERATE I
    END
  /* NOTHING WAS STACKED, MIGHT BE A LOCAL USERID */
  CPS TRANSFER CL 1 FROM NICKNAME
  PULL; PULL   /* CLEAN STACK AFTER CPS */
  IF RC=0 THEN SAY NICKNAME 'IS A LOCAL VM ID'
           ELSE SAY NICKNAME 'IS AN UNKNOWN NAME'
  END /* I */
EXIT

TELL:  /* TELL ABOUT THE PROGRAM */
  SAY 'CORRECT FORM:  ADDR NAME1 <NAME2 <NAME3 ....>>'
  SAY
  SAY 'ADDR SEARCHES YOUR RMSG FILE FOR THE SPECIFIED NICKNAME.'
  SAY 'IF IT FINDS THE NAME, IT DISPLAYS THE ACTUAL SYSTEM AND USERID'
  SAY 'OF THE USER.  IF THE NAME IS NOT FOUND, IT CHECKS FOR A LOCAL'
  SAY 'USERID WITH THE SAME NAME.'
```

SEND EXEC (FROM THE EXEC 2 DOCUMENTATION):

```
/* SEND FILE TO A LOCAL USER */
ARGS ENAME NAME FN FT FM Z
IF NAME='' | NAME='?' THEN DO
  SAY 'COMMAND IS: SEND USER FILENAME FILETYPE <FILEMODE>'
  EXIT 100; END
IF FT='' | Z¬='' THEN DO      /* CHECK NUMBER OF ARGUMENTS */
  SAY 'BAD SEND COMMAND'
  EXIT 101; END

IF FM='' THEN FM='*'          /* ASSUME ANY IF NO MODE GIVEN */
CP SPOOL PUN NAME CLASS A
IF RC¬=0 THEN DO              /* CHECK SPOOL WORKED */
  SAY NAME 'IS NOT A VALID USERID'
  EXIT 102; END
PUNCH FN FT FM
IF RC¬=0 THEN DO              /* CHECK PUNCH WORKED */
  SAY 'ERROR' RC 'FROM "PUNCH" (WHILE IN SEND)'
  NN=102
  END
 ELSE /* TELL RECIPIENT WHAT HAS BEEN DONE */
  CP MSG NAME 'I HAVE JUST PUNCHED YOU MY FILE' FN FT FM
CP SPOOL PUN '*' CLASS A
EXIT NN
```

SAMPLE EDITOR MACRO: EDITOR SUBCOMMANDS ARE SHOWN IN UPPER CASE FOR CLARITY.

```
/* REX EQUIVALENT OF CONC XEDIT (EXEC 2) MACRO */

/* FIRST COMPREHENSIVELY CHECK THE OPERANDS */
ARGS ENAME NUM FILL
SELECT
 WHEN NUM='' ; DO
    NUM=1; FILL=' '
    END
 WHEN FILL='' ; DO
    IF NUM='?' THEN DO
      /* CONC ?  */
      MSG '+-----------------------------------------+'
      MSG '|       CORRECT FORM IS: CONC <N <FILL>>  |'
      MSG '+-----------------------------------------+'
      MSG
      MSG 'CONCATENATE THE NEXT N LINES USING FILL STRING AS A '
      MSG 'SEPARATOR. DEFAULTS ARE 1 LINE AND SINGLE BLANK FILL.'
      MSG 'IF FILL=* THE LINES ARE TO BE CONCATENATED WITHOUT  '
      MSG 'ANY SEPARATORS.'
      EXIT; END
    FILL=' '
    END
 OTHERWISE
    IF FILL='*' THEN FILL=''
 END /* SELECT */

 IF DATATYPE(NUM)¬='NUM' THEN DO
    EMSG INVALID LINE COUNT "||NUM||"
    SIGNAL DISASTER; END

/* NOW CHECK IF THE CONCATENATED LINE WILL FIT IN THE FILE */
 TRANSFER LENGTH TRUNC LINE
 PULL L TRUNC FLINE
 IF L>255 THEN DO
    EMSG 'FILE TOO WIDE TO USE THIS MACRO'
    EXIT; END
 STACK 1 1 L
 PULL CURLINE
 STRING=CURLINE
 DO NUM
    NEXT
    IF RC¬=0 THEN DO
      EMSG EOF 'REACHED BEFORE CONCATENATING' NUM 'LINES.'
      GOTO  FLINE
      SIGNAL DISASTER; END
    TRANSFER LENGTH
    PULL LEN
    STACK 1 1 LEN
    PULL CURLINE
```

```
    STRING=STRING || FILL || CURLINE
    I=LENGTH(STRING)
    IF I>TRUNC THEN DO;
      EMSG 'CONCATENATED LINE LENGTH' I 'EXCEEDS TRUNC COLUMN' TRUNC
      GOTO FLINE
      SIGNAL DISASTER; END
    END; /* NUM */

  /* PUT THE CONCATENATED LINE IN THE FILE */
    GOTO FLINE
    REPLACE STRING
    NEXT
    DELETE NUM
    UP
    EXIT

DISASTER:
    ARGS PROMPT
    REPLY PROMPT
```

6. ACKNOWLEDGEMENTS

THE INSPIRATION FOR REX HAS MAINLY COME FROM THE STANDARD (CMS) EXEC LANGUAGE: MANY OF THE FEATURES FOLLOW DIRECTLY ON FROM THIS. MANY LANGUAGES HAVE INFLUENCED THE DEVELOPMENT OF REX - FOR EXAMPLE THE FLOW-CONTROL CONSTRUCTS ARE VERY PL/I LIKE, AS IS MUCH OF THE NOTATION; HOWEVER THE CONCEPT OF THE THE 'BLANK' OPERATOR WHICH CONCATENATES AND INSERTS A BLANK IS I BELIEVE ORIGINAL (PLEASE TELL ME IF IT IS NOT).

EXEC 2 (BY C.J.STEPHENSON), TOGETHER WITH THE YORKTOWN SVC PACKAGE (M.H.HACK), HAVE STRONGLY INFLUENCED THE LANGUAGE: PARTICULARLY IN THE AREA OF HOST DEPENDENCIES AND INTERFACES. THE "ADDRESS" INSTRUCTION SYSTEM' INTERFACE, FOR EXAMPLE, IS SIMILAR IN EFFECT TO THE EXEC 2 "&PRESUME" STATEMENT.

VERY MANY (AT LEAST TWO HUNDRED) PEOPLE HAVE MADE CONSTRUCTIVE CRITICISMS AND COMMENTS ON THE REX LANGUAGE; AND MANY HAVE CONTRIBUTED CODE AND DOCUMENTATION. MEMBERS OF THE REX LANGUAGE COMMITTEE ESPECIALLY HAVE BEEN OF CONSIDERABLE HELP IN THE DECISIONS LEADING TO RELEASE 2.01 OF THE REX PACKAGE.

THERE ARE NOW FAR TOO MANY TO GIVE THE INDIVIDUAL THANKS I WOULD LIKE TO HAVE INCLUDED IN THIS DOCUMENT, BUT ALL REX USERS ARE INDEBTED TO THOSE PEOPLE FROM ALL OVER THE COMPANY WHO HAVE CONTRIBUTED HELP, SUGGESTIONS, AND TIME.

MFC. 17TH AUGUST 1982.

APPENDIX: THE SUBCOMMAND CONCEPT

A SUBCOMMAND ENVIRONMENT USUALLY CORRESPONDS TO AN INTERACTIVE ENVIRONMENT, I.E. AN ENVIRONMENT IN WHICH A USER MAY ENTER COMMANDS TO BE EXECUTED IN THAT ENVIRONMENT. AN EXAMPLE IS AN EDITOR, WHICH ACCEPTS COMMANDS TO CHANGE, INSERT OR DELETE DATA IN A FILE, OR TO CHANGE THE CURRENT LOCATION IN A FILE. TO DISTINGUISH COMMANDS ISSUED TO A PARTICULAR ENVIRONMENT (SUCH AS AN EDITOR) FROM COMMANDS ISSUED DIRECTLY TO THE HOST (CMS), THE WORD 'SUBCOMMAND' IS USED.

INTERACTIVE USERS REACT TO THE SUCCESS OR FAILURE OF A PARTICULAR SUBCOMMAND BY ADAPTING AN INTENDED SEQUENCE OF COMMANDS. THEY ENQUIRE ABOUT SPECIFIC ATTRIBUTES OF THE ENVIRONMENT (E.G. LENGTH OF THE CURRENT LINE) AND BASE SUBSEQUENT SUBCOMMANDS ON THE INFORMATION SUPPLIED BY THE ENVIRONMENT (E.G. DISPLAYED IN A MESSAGE AREA).

THE SUBCOM MECHANISM MAKES THIS MODE OF INTERACTION AVAILABLE TO PROGRAMS AS WELL AS HUMAN USERS. IT GIVES PROGRAMS THE ABILITY TO ISSUE SUBCOMMANDS TO THE ENVIRONMENT, TO REACT TO THE OUTCOME OF A SUBCOMMAND, AND TO RETRIEVE INFORMATION ABOUT THE ENVIRONMENT FOR SUBSEQUENT USE.

TO USE THE SUBCOM MECHANISM, AN INTERACTIVE PROGRAM DECLARES A SUBCOMMAND ENVIRONMENT. THIS INVOLVES DECLARING THE NAME OF THE ENVIRONMENT, AND THE ENTRY POINT IN THE INTERACTIVE PROGRAM THAT IS PREPARED TO HANDLE SUBCOMMANDS ISSUED FROM OTHER PROGRAMS TO THE DECLARED ENVIRONMENT.

PROGRAMS WHICH ISSUE SUBCOMMANDS TO INTERACTIVE ENVIRONMENTS ARE OFTEN WRITTEN IN A CONVENIENT INTERPRETIVE LANGUAGE (SUCH AS EXEC 2 OR REX), AND ARE TRADITIONALLY CALLED MACROS. BOTH REX AND EXEC 2 HAVE THE CONVENTION THAT, UNLESS INSTRUCTED OTHERWISE, THEY DIRECT COMMANDS TO A SUBCOMMAND ENVIRONMENT WHOSE NAME IS THE FILETYPE OF THE MACRO. TRADITIONALLY, EDITORS DECLARE THEIR SUBCOMMAND ENVIRONMENT UNDER THEIR OWN NAME, AND CLAIM THAT NAME AS THE FILETYPE TO BE USED FOR THEIR MACROS.

FOR EXAMPLE, THE XEDIT EDITOR ('NEW CMS EDITOR' OF VM/SP, ANNOUNCED AT THE END OF JANUARY 1980) SETS UP A SUBCOMMAND ENVIRONMENT NAMED XEDIT, AND THE FILETYPE FOR XEDIT MACROS IS ALSO XEDIT.

THE MACRO ISSUES SUBCOMMANDS TO THE EDITOR (E.G. NEXT 4, OR TRANSFER ZONE). THE EDITOR 'REPLIES' WITH A RETURN CODE, AND SOMETIMES WITH FURTHER INFORMATION, WHICH IS STACKED, AND MAY BE READ BY THE MACRO. A NON-ZERO RETURN CODE FROM NEXT 4 MAY INDICATE THAT END-OF-FILE HAS BEEN REACHED, AND TRANSFER ZONE MAY STACK TWO NUMBERS, WHICH ARE THE CURRENT SETTING OF THE 'ZONE' IN XEDIT. BY TESTING THE RETURN CODE AND RETRIEVING STACKED INFORMATION, THE MACRO HAS THE ABILITY TO REACT APPROPRIATELY, AND THE FULL FLEXIBILITY OF A PROGRAMMABLE INTERFACE IS AVAILABLE.

REX ALLOWS THE DEFAULT ENVIRONMENT TO BE ALTERED (BETWEEN VARIOUS SUBCOMMAND ENVIRONMENTS OR THE HOST EVIRONMENT) USING THE ADDRESS

STATEMENT. EXEC 2 HAS A SIMILAR MECHANISM IN THE &PRESUME STATEMENT.

THE SUBCOM COMMAND IS USED TO DECLARE, QUERY, OR CANCEL SUBCOMMAND ENVIRONMENTS.

ONLY THE QUERY FORM OF SUBCOM IS A COMMAND, IN THE SENSE THAT IT CAN BE ISSUED FROM THE TERMINAL (OR FROM AN EXEC FILE). THE FORM OF THIS COMMAND IS:

SUBCOM NAME

THIS YIELDS A RETURN CODE OF 0 IF 'NAME' IS CURRENTLY DEFINED, OR 1 IF IT IS NOT DEFINED AS A SUBCOMMAND ENVIRONMENT NAME.

PROGRAMS MAY CALL THE SUBCOM FUNCTION WITH AN APPROPRIATE PLIST TO DECLARE OR CANCEL AN ENVIRONMENT, OR TO OBTAIN COMPLETE INFORMATION ABOUT A DECLARED ENVIRONMENT. THE PLISTS ARE DEFINED IN YKTSVC MEMO. (A 'FUNCTION' TAKES A PARAMETER LIST WHICH MAY CONTAIN BINARY INFORMATION, SUCH AS FLAGS OR BINARY ADDRESSES, AND IS THUS DISTINGUISHED FROM A 'COMMAND', WHICH TAKES CHARACTER STRING ARGUMENTS ONLY.)

THE COMMAND SUBMAP CAN BE USED TO LIST CURRENTLY DEFINED SUBCOMMAND ENVIRONMENTS.

(FROM SUBCOM MEMO BY MICHEL HACK, YORKTOWN HEIGHTS, FEBRUARY 1980)

--- END OF DOCUMENT ---